

AD-A145 872

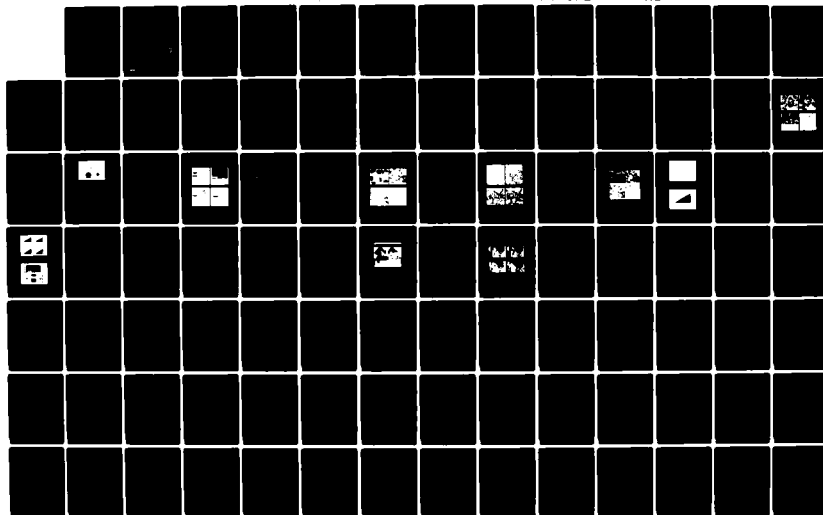
TEST-BED FOR PROGRAMMABLE AUTOMATION RESEARCH PHASE I
(U) SRI INTERNATIONAL MENLO PARK CA R C SMITH ET AL.
APR 84 AFOSR-TR-84-0776 F49620-82-K-0034

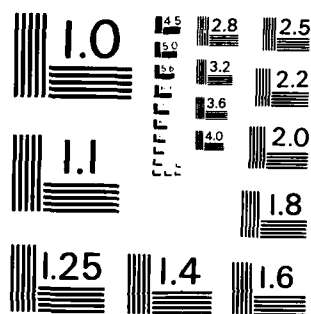
1//

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

11

TEST-BED FOR PROGRAMMABLE AUTOMATION RESEARCH

Final Report—Phase I

April 1984

By: R. Smith, R. Bolles, J. Herson, J. Myers,
D. Nitzan, and W. Park
Robotics Laboratory

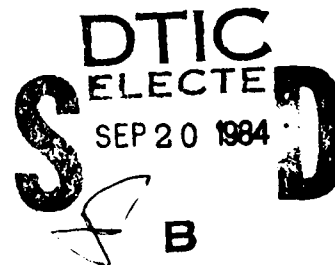
Prepared for:

Air Force Office of Scientific Research
Bolling Air Force Base, Washington, D.C. 20332

Attn: Dr. Alan H. Rosenstein, Program Manager
Electronic and Material Sciences

Contract F49620-82-K-0034

SRI Project 4703



AFOSR-DR-84-0917011

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025-3493
(415) 326-6200
Telex: 334486



84 09 17 011

AD-A145 872

DTIC FULL COPY

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS none		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) SRI Project 4703			5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR		
6a. NAME OF PERFORMING ORGANIZATION SRI International		6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U. S. Air Force		
6c. ADDRESS (City, State and ZIP Code) 333 Ravenswood Avenue Menlo Park, California 94025			7b. ADDRESS (City, State and ZIP Code) Office of Scientific Research Building 410 Bolling Air Force Base, DC 20332		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U.S. Air Force		8b. OFFICE SYMBOL (If applicable) NE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F49620-82-K-0034		
8c. ADDRESS (City, State and ZIP Code) Office of Scientific Research Building 410 Bolling Air Force Base, DC 20332			10. SOURCE OF FUNDING NOS		
11. TITLE (Include Security Classification) Test-Bed for Programmable Automation Research			PROGRAM ELEMENT NO. 61102F	PROJECT NO. 2305	TASK NO. K1
12. PERSONAL AUTHOR(S) Smith, R., Bolles, R., Herson, J., Myers, J., Nitzan, D., Park,					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM 9/1/82 TO 3/31/84		14. DATE OF REPORT (Yr., Mo., Day) 1984, April	
15. PAGE COUNT 101					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary; and identify by block number)		
FIELD	GROUP	SUB. GR.			
19. ABSTRACT (Continue on reverse if necessary and identify by block number) In the 18-month project sponsored by AFOSR under Contract F49620-82-K-0034, SRI performed six research and development tasks: We have developed methods for improving the accuracy with which a computer can estimate the uncertainty in spatial relationships among objects; we have acquired and interfaced a commercial triangulation range sensor in order to investigate the use of range data in determining part location, and have implemented a method capable of identifying and eliminating spurious data caused by commonly occurring secondary reflections; we have selected, implemented, and characterized a set of low-level edge-operators and feature detectors for use in programmable assembly; and, in two related efforts, we have developed methods for collision-avoidance and coordination of systems containing multiple manipulators. SRI's programmable assembly test-bed has been extended to aid this development, and has been the vehicle for demonstration of the results.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS <input type="checkbox"/>			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Alan H. Rosenstein			22b. TELEPHONE NUMBER (Include Area Code) 202-767-4984		22c. OFFICE SYMBOL NE

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

EXECUTIVE SUMMARY

The Air Force Office of Scientific Research sponsored an 18-month project with SRI International, entitled "Test-Bed for Programmable Automation Research," whose goals were to

- Explore and develop programmable automation techniques for robot manipulation, sensing, and industrial vision.
- Extend our programmable assembly station as a test-bed for generating and demonstrating those techniques.

Five research tasks were concerned with two general problems: determining the location of parts, and moving a manipulator safely, without collisions. A sixth task was the extension of SRI's programmable assembly station.

Task 1: Analysis of Locational Uncertainty

We have developed methods for improving the accuracy with which a computer can estimate the uncertainty in spatial relationships among objects. The uncertainty arises from the use of inaccurate manipulators, part feeders, and sensors. An automatic planner that defines a task sequence using these devices must be able to reason about the uncertainties they introduce in order to select the assembly strategies that can succeed despite potential variations. Our methods for estimating uncertainty are believed to be unique in their use of probability distributions to describe the variables of a spatial relationship, rather than maximum and minimum values only. We have produced formulae that describe this uncertainty and that may be quickly computed.

Task 2: Acquisition and Analysis of Range Data

We have acquired and interfaced a commercial triangulation range sensor (Technical Arts' White Scanner Model 100A) to our VAX-11/750 in order to investigate the use of range data in determining part location. A problem common to such sensors is the pickup of spurious data from secondary reflections. We have implemented a technique for detecting and eliminating data caused by these reflections.

Task 3: Characterization of Feature Detectors

We have selected, implemented, and characterized a set of low-level edge-operators and feature detectors, and made some improvements in them. The characterizations will be useful for automatic selection of part-location strategies--a future goal.

Task 4: Multiarm Collision Avoidance

We have developed a system that performs general multiarm collision avoidance in near real time, using an approach whereby a simulated potential field is placed around the arms and obstacles. We have investigated alternative methods of implementing a potential field collision-avoidance system, with associated tradeoffs between computational effort and accuracy. We have implemented a unique system that considers the entire volume of the arm for collision avoidance, rather than a few discrete points. The system operates with multiple arms simultaneously, allows arms to pick up objects, and performs avoidance with the arm and held object. The system has been demonstrated on our assembly station using two PUMA 560 manipulators and numerous obstacles.

Task 5: Coordination of Multiple Manipulators

In a different approach to the collision-avoidance problem, a state-space (joint-space) representation of a system of manipulators was developed. An obstacle map was formed in this space, which was compressed using the CMAC algorithm by Albus [1]. As a side-effect, transitions from "obstacle" to "no obstacle" in the binary-valued map were smoothed; the graded map value was defined to be "distance to the obstacle" and a potential field method of collision avoidance developed with this data. The method presents a unified approach to a number of manipulator control problems.

Task 6: Extension of a Programmable Assembly Station

We have made extensions to an existing flexible assembly station composed of multiple robots and sensors. Our scheme for message-based control of the station devices was reimplemented on new Ethernet hardware replacing the previous local-area network. A VAX-11/730 was installed to coordinate the station's activities. New devices, such as a graphic display, image processor, and range-finder have been acquired and interfaced to the station or to a VAX-11/750. Some of the described work was demonstrated with this assembly station.

CONTENTS

LIST OF ILLUSTRATIONS	ix
I INTRODUCTION	1
II ANALYSIS OF LOCATIONAL UNCERTAINTY	5
A. Introduction	5
B. Motivation	5
C. Comparison to Prior Work	6
D. Method of Approach	7
1. Representation of Fuzzy Relationships	7
2. Computing the Product of Two Fuzzy Transformations	10
3. Evaluation	12
E. Uses of Locational Uncertainty Analysis	13
F. Future Work	16
III RANGE DATA ACQUISITION	17
IV CHARACTERIZATION OF FEATURE DETECTORS	25
A. Zero Crossings	27
B. Line and Arc Fitting	33
V MULTIARM COLLISION AVOIDANCE	37
A. What Is Collision Avoidance?	37
B. The Difference Between Collision Detection and Collision Avoidance	38
C. The Motivation for Collision Avoidance	38

D.	Other Approaches	39
E.	Accomplishments	40
F.	Description	40
G.	The Potential-Field Approach	41
H.	An Example	42
I.	Advantages and Disadvantages of the System	44
J.	Remaining Areas of Research	45
VI	COORDINATION OF MULTIPLE MANIPULATORS	47
A.	Introduction	47
B.	Motivation	47
C.	Method of Approach	47
1.	Collision Map in State Space	48
2.	Collision Map Compression	50
3.	Uses for a Collision Map	50
4.	Comparison to Work by Others	52
D.	Future Work	53
VII	PROGRAMMABLE ASSEMBLY TEST-BED	55
APPENDICES		
A	FORMULAE FOR PRODUCT OF TWO NOMINAL TRANSFORMATIONS	61
B	FUZZY TRANSFORMATION PRODUCT: USING GAUSS'S LAW	65
C	FUZZY TRANSFORMATION PRODUCT: USING SPHERICAL GEOMETRY MODEL	77
D	ALGORITHMS FOR COLLISION AVOIDANCE	83

ILLUSTRATIONS

1	Spatial Relationships in a Task	9
2	Computing Robot/Block Relationship	9
3	Products of Two Fuzzy Transformations	10
4	Product of Transformations in Spherical Coordinates . . .	13
5	Sample of Estimation vs Monte Carlo Results	14
6	White Scanner Data	18
7	Z Image With Incorrect Data	20
8	Diagram of a Secondary Reflection	20
9	A Sequential Pair of Secondary Reflections	21
10	Filtering Out Secondary Reflections	22
11	An Example of an Image Reversal	23
12	Typical Steps for Recognizing an Object	26
13	Edges Detected With Different-Sized Kernels	28
14	Edges Passing Different Magnitude Thresholds	30
15	Graph of the Number of Accepted Edges as a Function of the Magnitude Threshold	31
16	An Example of the Displacement of Edges Caused by Large Kernel Sizes	31
17	Arc and Line Fitting With Different Parameter Settings .	34
18	Another Example of Arc and Line Fitting	34
19	Model of the Assembly Station	41
20	Simultaneous Collision Avoidance with Two Robot Arms	43

21	State Variables for Two Manipulators	49
22	Two-Dimensional Cross Sections of the Collision Map in State Space	49
23	Collision Map Cross Section After Compression and Smoothing by CMAC	51
24	Assembly Station Block Diagram	56
C-1	Components in the Estimation of Angular Variances . . .	79

I INTRODUCTION

The Air Force Office of Scientific Research sponsored an 18-month program at SRI International, with principal emphasis on research and development of robotic assembly, inspection, and rework capabilities. SRI's goals in this project were to

- Explore and develop programmable automation techniques for robot manipulation, sensing, and industrial vision.
- Extend our programmable assembly station as a test-bed for generating and demonstrating those techniques.

To meet these goals, six tasks were specified:

- (1) Analysis of Locational Uncertainty
- (2) Range Data Acquisition
- (3) Characterization of Feature Detectors
- (4) Multiarm Collision Avoidance
- (5) Coordination of Multiple Manipulators
- (6) Programmable Assembly Test-Bed.

Five of the six tasks can be classed into two broad categories: locating objects in the work environment and maneuvering multiple arms safely to and from destinations (such as the located parts) without collisions among themselves or other obstacles. The sixth task is the extension of our programmable assembly station. SRI's long-range interests are in the development of automatic or semi-automatic systems to produce assembly programs; these systems will necessarily rely on the techniques being developed.

Objects must be located in order to perform operations with them; but, depending upon the operation, the precision of information required can be very different. The two operations, "pick up a small, steel part with a magnet," or "pick up the same part with a hand," impose vastly different constraints on how well we must know the part's location to succeed. Knowledge of spatial relationships in an assembly task is inherently uncertain because of positioning errors by manipulators and feeders, measurement errors by sensors, and tolerances in the dimensions of a part. Often, even small locational errors can become magnified by a sequence of manipulations in an assembly task. To plan an assembly task, we must be able to reason about these uncertainties and how they are compounded--this is the problem addressed by the first task. If we can estimate the uncertainty in the part's location, we can answer the question "Will the planned operation succeed, despite the (estimated) variability?"

When the answer is "no," one possible solution is to improve our knowledge of the part's location by planning a sensing step. A flexible automation system that is to perform a wide variety of tasks must be able to acquire and process different types of data. One type that has been somewhat neglected by the research community is range data. This neglect is mainly due to the lack of suitable rangefinders. Commercial devices are now becoming available, and part of SRI's research effort has been directed toward the acquisition and analysis of range data for locating objects.

Strategies for locating an object generally consist of selecting a sequence of low-level processing techniques to detect features and a set of matching techniques to compare test objects with models. In order to select feature detectors and matching techniques, an automatic programming system must know what the choices are, when they can be applied, and what results they are likely to produce. That is, the system needs characterizations of the techniques in a form it can use to list the possible ways of achieving a goal and to compare the possibilities according to their cost and reliability. Part of SRI's effort has been directed at selecting, implementing, and characterizing a set of edge operators and feature detectors as a step toward an automatic programming system.

In many cases, a part is being located for acquisition by a manipulator. If the location of the part may vary widely, the trajectory and locations of the manipulator acquiring it will also vary. When two or more manipulators cooperate in an assembly task, their motions must be coordinated so that they do not collide with each other or their surroundings. However, the interactions of the separate arm trajectories and object geometries are complicated, which makes manual determination of arm trajectories difficult and subject to error. An automatic collision-avoidance system is needed to ensure that collision-free trajectories are computed for the arms. In separate efforts, we have investigated two methods for collision avoidance and coordination of multiple manipulators, the first method using a Cartesian coordinates space and the other using an arm-joint-coordinates space (or state space).

The final component of the AFOSR project is the extension of the SRI Programmable Assembly Test-Bed. The test-bed is an assembly station, which serves to integrate much of our work and supports experimentation. We have added a number of new capabilities to the station under the AFOSR project, and used it in developing and demonstrating our research activities.

II ANALYSIS OF LOCATIONAL UNCERTAINTY *

A. Introduction

We describe methods for improving the accuracy with which a computer can estimate the uncertainty in spatial relationships among objects, manipulators and sensors. Such methods will become important in the off-line development and simulation of assembly tasks.

B. Motivation

Knowledge of spatial relationships among the objects in an assembly task is inherently uncertain, because of positioning errors by manipulators and feeders, measurement errors by sensors, and tolerances in the dimensions of a part. Often, even small locational errors can be magnified by a sequence of manipulations in an assembly task. Thus, it may be difficult, as well as expensive, to minimize this locational uncertainty through the use of special-purpose fixtures and high-precision instruments. A program for a flexible assembly station may use sensing and an intelligent selection of strategies as an alternative, more general solution.

In developing such a program, an automatic planner (or person) will have to estimate the uncertainty in the spatial relationships of objects defined in the task in order to select assembly strategies capable of accommodating the potential variation. The planner must also decide when and how to use sensing to reduce uncertainty, so that available strategies can be applied.

* By Randall C. Smith

C. Comparison to Prior Work

Taylor [23] described the importance of determining and accommodating the uncertainties in part locations when developing programs for robotic assembly tasks. He applied some linear programming techniques to the analysis of locational uncertainty and described potential uses of the analysis, one of which was planning for measurements. Brooks [2] developed a means of bounding the uncertainty in a part's location symbolically, and was interested in sensor planning using the result of the analysis.

However, the methods developed by Taylor and Brooks for analyzing locational uncertainty used computations based on the *maximum* and *minimum* values of the variables describing spatial relationships. The result can be a very conservative estimate of the uncertainty in a relationship that is computed from a chain of other relationships.

When a task sequence is planned involving uncertain spatial relationships, strategies that can accommodate the anticipated variation are selected to accomplish the task. By overestimating the potentially applicable strategies may be overlooked.

To improve accuracy beyond that possible when using maximum and minimum bounds on variables, SRI has explored the use of probability distributions to represent the variables describing a relationship. We believe the use of probability distribution information in this application to be unique. We have developed two unique and different sets of formulae for estimating the variability in the spatial relationship between two object coordinate frames. We have also implemented a Monte Carlo sampling approach to determine the variability, and use it as a check for the estimation formulae.

Methods for developing these estimates of locational uncertainty are first described, followed by an example of their potential use.

D. Method of Approach

Spatial relationships among objects are defined as the relationships among coordinate frames defined in the objects. These relationships are never known exactly; however, by modeling our positioning and measurement devices, we assume we can characterize the variations they contribute when utilized to create or measure a relationship. We define *fuzzy transformations* to describe the expected relationship and the expected variation. In a graph of fuzzy transformations among objects, some relationships are given, others must be computed from a chain of intermediate relationships. Our methods described will compute the compound fuzzy transformation by computing the product of two fuzzy transformations, using the result and the next relationship in the chain to compute a new result, and so on, until the end of the chain is reached. Thus, the fuzzy relationship between any two coordinate systems connected in the graph can be computed.

1. Representation of Fuzzy Relationships

Objects (workpieces, manipulators, sensors) will be assumed to have a coordinate frame defined in them, so that the nominal spatial relationship between two objects can be described as the relationship between the related coordinate frames. This relationship is represented by a transformation that translates and then rotates one coordinate frame into the other. Six degrees of freedom are represented--three components of the frame's translation (x, y, z) and three rotations (ϕ, θ, ψ) in a specified order about the coordinate frame's axes. The formulae to be described are derived from an Euler angle representation (though other representations would work as well), defined as

- (1) Rotate ϕ radians about the z-axis
- (2) Rotate θ radians about the new y-axis
- (3) Rotate ψ radians about the new z-axis.

These transformations are generally used as if they represented the *actual* spatial relationship between two coordinate frames; but of course there is always some unknown error. In a flexible assembly environment where sensing and

intelligence are supposed to replace error-reducing fixturing, the unknown errors may be quite large, and cannot be ignored when assembly plans are defined. The potential variation in a spatial relationship must be characterized.

We assume we can model the assembly equipment, and make use of information about its characteristic positioning and measurement errors, when developing a program for an assembly task. Thus, when a manipulator positions and orients a part, it contributes some error, and we have some knowledge about this error.

We define a *fuzzy transformation* as a transformation relating two coordinate frames in which the degrees of freedom are treated as random variables. A fuzzy transformation has twelve components--the six nominal values of the transformation $(x, y, z, \phi, \theta, \psi)$ and variance values for each. The variance values will either be given in the models of the devices with which we plan to create and measure locations, or computed.

Transformations are often compounded--relating, for example, the coordinate system of a block on a conveyor to the coordinate system of a robot (see Figure 1). The potential variation in the relationship between the block and robot is a complicated function of the potential variations in the intermediate defining relationships (e.g., robot to feeder, feeder to conveyor).

Computing a compound fuzzy transformation is the basic task of this work. Given a chain of fuzzy transformations, the compound result will be computed from two transformations at a time; i.e., the result of the product of the first two fuzzy transformations will be used with the next fuzzy transformation in the product chain to get a further result, until the end of the chain is reached and the overall result computed (see Figure 2). Thus, the problem is to estimate the result of a product of two fuzzy transformations. There are four distinct ways in which the two fuzzy transformations may be combined, shown in Figure 3, with the dashed line indicating the desired relationship to be computed. Only formulae dealing with the first combination shown have been derived; formulae for the other combinations necessary are under development, and are derived similarly.

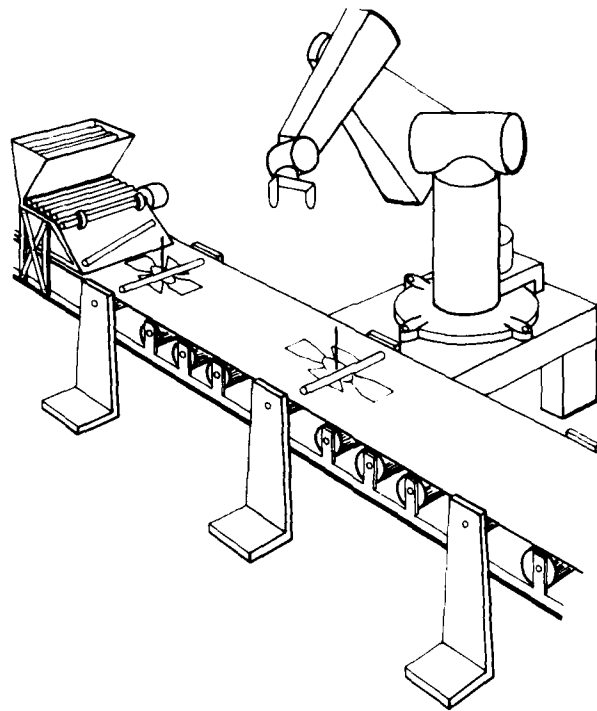


FIGURE 1 SPATIAL RELATIONSHIPS IN A TASK

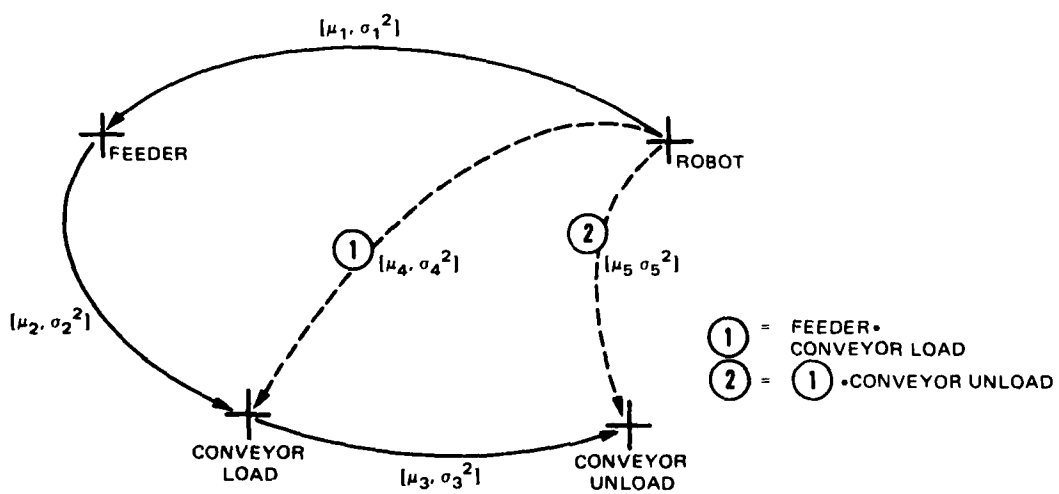
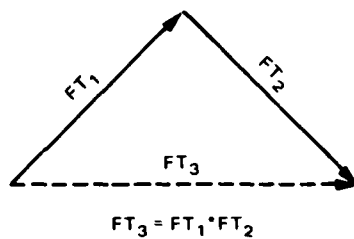
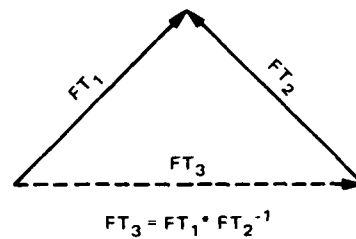


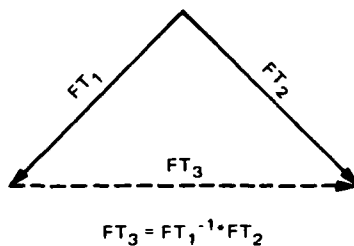
FIGURE 2 COMPUTING ROBOT/BLOCK RELATIONSHIP



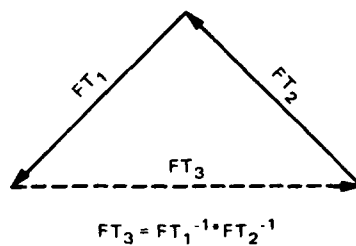
(a)



(b)



(c)



(d)

FIGURE 3 PRODUCTS OF TWO FUZZY TRANSFORMATIONS

2. *Computing the Product of Two Fuzzy Transformations*

In the following we will make use of the formulae for the product of two nominal transformations (all variances are zero),

$$T_3 = T_1 * T_2.$$

(We will use this shorthand notation for the operation of computing the compound transformation.) Isolating, for example, x_3 as a function of the variables in T_1 and T_2 , we write

$$x_3 = \text{FNX3}(x_2, y_2, z_2, x_1, \phi_1, \theta_1, \psi_1)$$

where the variables have subscripts matching the transformation with which they are associated. x_3 is not a function of the angles from T_2 or of y_1 and z_1 from

T_1 . The functions for $x_3, y_3, z_3, \phi_3, \theta_3$, and ψ_3 are readily derived from the product $T_1 * T_2$ [17] and are given in Appendix A.

Three methods presented below can be used to compute FT_3 as the product of fuzzy transformations FT_1 and FT_2 (as shown in Figure 3a). Following a description of the three methods is an evaluation.

a. Monte Carlo Simulation

The Monte Carlo simulation method will use the functions for the product of nominal transformations given in Appendix A. Values of each of the 12 variables from FT_1 and FT_2 are chosen at random based on their given mean and variance values, and the six results $[x_3, y_3, z_3, \phi_3, \theta_3, \psi_3]$ are computed. A running mean and variance calculation is performed for each result as the input is sampled. When a sufficiently large number of samples have been taken, the final mean and variance values of the outputs represent the fuzzy transformation FT_3 . This particular method provides a useful check for our estimation formulae described below, because it is straightforward to implement and its accuracy is determined by the size of the sample used. Unfortunately, a sample size of 1 million is necessary to produce an answer accurate to three decimal places, and the resulting program is time consuming to run.

b. Gauss's Law

This method uses Gauss's Law for the Propagation of Errors in determining the variances of the variables of FT_3 . The law states that given a function of random variables, e.g.,

$$x_3 = \text{FNX3}(x_2, y_2, z_2, x_1, \phi_1, \theta_1, \psi_1),$$

the variance of the function FNX3 is approximated by

$$\sigma^2_{\text{FNX3}} \approx (\partial \text{FNX3} / \partial x_2)^2 * \sigma^2_{x_2} + \dots + (\partial \text{FNX3} / \partial \psi_1)^2 * \sigma^2_{\psi_1}.$$

The partial derivatives of FNX3 are evaluated at the given mean values of the variables from FT_1 and FT_2 .

The mean values of the variables of FT_3 are approximated by the values of the product $T_1 * T_2$ (where T_1 corresponds to FT_1 above, and T_2 to FT_2 , but with all variances $= 0$).

The formulae derived by this method are presented in Appendix B. The formulae are complicated, although they have undergone considerable algebraic simplification.

c. Spherical Geometry Model

An alternative set of formulae to estimate variances in the angles of FT_3 (ϕ_3 , θ_3 , and ψ_3) has been developed by the author, based on a geometric argument in which the Euler angles of FT_1 and FT_2 are represented as two oriented points in a spherical coordinate system. Together with the point (0,0,0) they form a spherical triangle, as shown in Figure 4. The formulae for the variance of the angles in FT_3 require computation of lengths of sides of right spherical triangles. The derivation is too detailed to be presented here and is being prepared for separate publication, but the resulting formulae for the variance in each of the three angles are surprisingly simple. They are given in Appendix C, along with their geometric interpretation.

3. *Evaluation*

Figure 5 shows sample output from a program that computes the product of two fuzzy transformations by Monte Carlo simulation and by the two different estimation formulae. These results are quite close in value and typical of the trial samples that have been run. However, most sample comparisons have been made between the estimation formulae only; only a few comparisons to the Monte Carlo result have so far been performed.

Notice that the standard deviation value given for the angles of FT_1 and FT_2 is 0.1 radians; thus, ± 4 standard deviations corresponds to an error distribution covering ± 23 degrees in our examples--a large range. The estimation formulae from either method can be quickly computed, with the results obtained in a fraction of second on a VAX-11/750. Both sets of formulae contain terms with denominators that approach zero as θ_3 approaches $n * \pi$. Behavior of the formulae around this singular region is being determined.

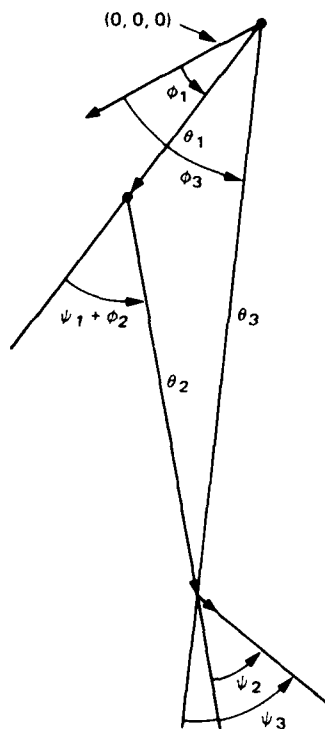


FIGURE 4 PRODUCT OF TRANSFORMATIONS
IN SPHERICAL COORDINATES

E. Uses of Locational Uncertainty Analysis

We believe that an accurate analysis of locational uncertainty can serve as an important criteria in making two categories of decisions in preplanning a robotic task:

- Selection of successful strategies that accommodate the estimated variation.
- Selection of sensors to reduce uncertainty to limits where available strategies are applicable.

In addition, we feel that locational uncertainty analysis can provide information useful in configuring a workstation for the planned task, by:

- Providing information for a sensitivity analysis of the errors, so that potential errors can be reduced during workstation design (by adding fixturing, or more precise equipment) if it is cost effective to do so.

Simulations based on 10,000 trials

GIVEN: (INPUT VAR: mean, sd)
 (X1 : 150.000 1.000) (Y1 : 50.000000 1.000000) (Z1 : 200.000000 1.0000)
 (PHI1: 0.5000 0.1000) (TH1 : 0.900000 0.100000) (PSI1: 0.700000 0.100000)

(X2 : 200.000 1.000) (Y2 : 100.000000 1.000000) (Z2 : -100.0000 1.0000)
 (PHI2: 0.4000 0.1000) (TH2 : 0.400000 0.100000) (PSI2: 0.700000 0.100000)

X3 Y3 Z3 PHI TH PSI

NOMINAL MEANS: 31.120680 219.025663 68.478015 0.895489 1.121674 1.586653
 M CARLO MEANS: 31.568873 217.086143 69.693102 0.895171 1.122054 1.589957

M CARLO SDs : 25.148872 13.760166 16.253671 0.137024 0.119886 0.161467
 GAUSS SDs : 25.680710 13.631217 16.313058 0.138703 0.119715 0.161562
 NEW SDs : ***** 0.138705 0.119900 0.161725

GIVEN: (INPUT VAR: mean, sd)
 (X1 : -100.000 1.0000) (Y1 : 200.0000 1.0000) (Z1 : 50.0000 1.000000)
 (PHI1: 1.3000 0.100000) (TH1 : 1.5000 0.10000) (PSI1: 0.200000 0.100000)

(X2 : -50.0000 1.0000) (Y2 : -50.0000 1.0000) (Z2 : 400.0000 1.0000)
 (PHI2: -0.5000 0.10000) (TH2 : -0.60000 0.1000) (PSI2: 0.4000 0.1000)

X3 Y3 Z3 PHI TH PSI

NOMINAL MEANS: 62.781243 566.029281 117.266872 1.509419 0.931720 0.023939
 M CARLO MEANS: 61.483417 562.254187 116.778258 1.510658 0.939261 0.021145

M CARLO SDs : 36.535755 17.910190 39.919777 0.148023 0.137059 0.204323
 GAUSS SDs : 36.878673 17.589822 40.082139 0.144551 0.138127 0.202409
 NEW SDs : ***** 0.144725 0.138188 0.201906

GIVEN: (INPUT VAR: mean, sd)
 (X1 : 300.0000 1.0000) (Y1 : 300.0000 1.0000) (Z1 : 200.0000 1.0000)
 (PHI1: 0.80000 0.10000) (TH1 : 0.70000 0.10000) (PSI1: 0.600000 0.10000)

(X2 : 100.0000 1.0000) (Y2 : -100.000 1.0000) (Z2 : -300.0000 1.0000)
 (PHI2: 0.60000 0.10000) (TH2 : 0.50000 0.10000) (PSI2: 0.400000 0.100000)

X3 Y3 Z3 PHI TH PSI

NOMINAL MEANS: 258.119603 219.460411 -118.997503 1.369292 0.977260 1.210044
 M CARLO MEANS: 257.483521 219.718139 -116.849314 1.365780 0.979302 1.212309

M CARLO SDs : 25.020041 25.583557 9.239953 0.149998 0.118547 0.159305
 GAUSS SDs : 25.188415 25.820985 8.968296 0.148786 0.119414 0.158700
 NEW SDs : ***** 0.148797 0.119638 0.158813

FIGURE 5 SAMPLE OF ESTIMATION vs MONTE CARLO RESULTS

- Providing information about sensing volumes necessary to locate parts, thus suggesting useful mounting locations for sensors.

Suppose we are to write a program for a robot workstation, to acquire a wooden block from a pallet and drop it into the middle of an open box. A prior plan calls for the block to be placed on the pallet by a low-precision device, and the pallet itself to be positioned in front of the manipulator, again with low-precision. If we can characterize the positioning errors of our devices, then our uncertainty analysis tool can be used to estimate the overall variability in the block's initial location with respect to the manipulator.

If we estimate the block to be very close to its planned nominal location, the pickup can be simply accomplished. If the variation in the block's orientation is small, but its variation in position significant, we might plan to "corral" the block by opening the gripper wide and capturing the block somewhere between the fingers as the gripper is closed. This strategy is acceptable because the final relationship between the block and gripper does not need to be well known; the block can be dropped into the open box no matter how it is grasped, if the arm positions itself above the box's center.

The estimated variation of the block about its nominally planned position can be used to decide if the gripper can be opened widely enough to capture the part. If not,

- (1) Plan to use another gripper, whose maximum opening can now be specified.
- (2) Plan to feed the block to the robot more accurately with other equipment.
- (3) Plan to introduce a sensing step to locate the part.

Suppose we choose Option 3 as "best" for our purposes by some cost measure. The chosen sensor need only locate the part well enough so that it can be corraled, as before. A camera can certainly achieve this level of accuracy in determining part location. We may choose to use a camera and a sensing strategy to locate the block, which requires that the block be entirely in view. Our

estimate of the variability in the block's location can be used to define the minimum field of view and position of the camera, ensuring that the entire block will be seen. Thus, we have decided to add a camera to the workstation as part of our plan for accomplishing the task, and we have a good idea where to position it, as well as some information about the lens to use.

Alternatively, we might have planned to use a much cruder sensor: A contact switch on the finger could feel for the side of the block, giving a rough estimate of its position. If we plan such a sensing step, we can assume that when it is executed, and the switch closed, we will know enough about the block's position to "corral" it, having reduced the uncertainty in the critical dimension to an acceptable level.

F. Future Work

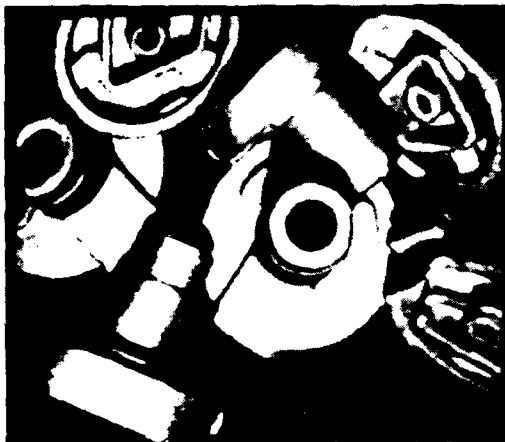
A quantitative evaluation of the performance of the estimation formulae remains to be attempted; use of the Monte Carlo simulation to check a large sample of the estimation results is impractical because of the computation time required. Other methods for evaluating performance will have to be devised. In addition, the formulae must be completed to account for the other possible forms (see Figure 3) in the product of two fuzzy transformations. Although this task is tedious, no problems are anticipated. A complete package to estimate the variability between any two given coordinate frames in a interconnecting set of object relationships is the immediate goal; future planning research anticipates the availability of such a package for the uses previously outlined.

III RANGE DATA ACQUISITION *

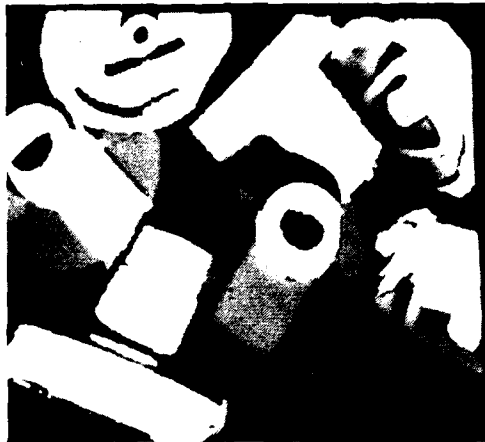
If a flexible automation system is to perform a wide variety of tasks, it must be able to acquire and process different types of data. One type that has been somewhat neglected by the research community is range data, primarily because of the lack of devices for acquiring such data. Recently, however, such companies as Robot Vision Systems Incorporated, Technical Arts Corporation, and Diffracto Ltd., have started producing range-measurement devices. In order to extend the class of tasks that our flexible automation test-bed could perform, we purchased a White Scanner Model 100A from Technical Arts Corporation in Seattle, Washington, and interfaced it to our VAX-11/750. With this device we are able to gather "dense" range images (200 x 240 pixels) in approximately 8 minutes. Figure 6 shows an example of the data produced by the device. The image in the upper left corner is an intensity image, which is registered with the range data. The image in the upper right corner is the z component of the range data, the height of the objects above the table. The x component is shown at the lower left. The y component is not shown. The image at the lower right is a code image returned by the White Scanner to indicate the status of each pixel. Codes are included for such things as no data, good data, blooming, and ambiguous data.

Since we received one of the first devices produced by Technical Arts, we hit several bugs in the system, including a bad laser positioning servo, a set of host-to-White Scanner communication problems, and a thresholding bug. The people at Technical Arts have worked closely with us to solve these problems. Another problem, which is common to all triangulation-based systems, is the occurrence of multiple reflections. In the next section we briefly describe an idea for detecting and eliminating them.

* By Robert C. Bolles and James H. Herson



(a) INTENSITY IMAGE



(b) Z IMAGE



(c) X IMAGE



(d) CODE IMAGE

FIGURE 6 WHITE SCANNER DATA

Multiple reflections of the light sheet cause problems for all structured light sensors that use a two-dimensional camera. The problem is that the system cannot distinguish primary reflections from secondary and tertiary reflections. The standard heuristic is to assume that the brightest reflection is the primary one. This heuristic often fails when the object surfaces are shiny. The White Scanner uses this heuristic. Another step it takes to avoid reporting bad data is to flag data computed from image rows containing more than one reflection. (One minor problem that Technical Arts is fixing is that at present they flag ambiguous rows, but don't return any data. Multiple interpretations sometimes arise from the the projector-object-camera configuration, and the data from one of the reflections would be useful.)

Sometimes only the secondary reflection is seen by the camera. In this case, the heuristic on brightness and the multiple-reflection flagging don't help. There is only one bright reflection, but it leads to erroneous data. Figure 7 shows an example of a dense image taken by the White Scanner in which some incorrect x-y-z values have been computed from secondary reflections. The image is a height image, in which the height above the table is encoded in gray scale. Higher points are brighter. A cylindrical object is standing on end in the lower left corner of the image. The heights on the "front" surface of the cylinder are wrong.

Figure 8 illustrates how these incorrect values were computed. The camera in Figure 8 sees only the secondary reflection of the light sheet, which appears on the table. It computes a height for that point as if it were on the light sheet (i.e., at point P). Because P is a reasonable point, it can't be recognized as being "bad" by just looking at its coordinates. However, if the light sheet is moved slightly to the right, the x coordinate is expected to stay the same or increase, except in a few special cases. However, as illustrated in Figure 9, the x coordinates computed from secondary reflections decrease. Therefore, by analyzing pairs of points it is possible to filter out data computed from secondary reflections. Figure 10 illustrates how this filtering is done. The raw data are in the upper left corner; the x coordinate is in the upper right corner; a mask of secondary reflections is in lower left corner; and the corrected data are shown at

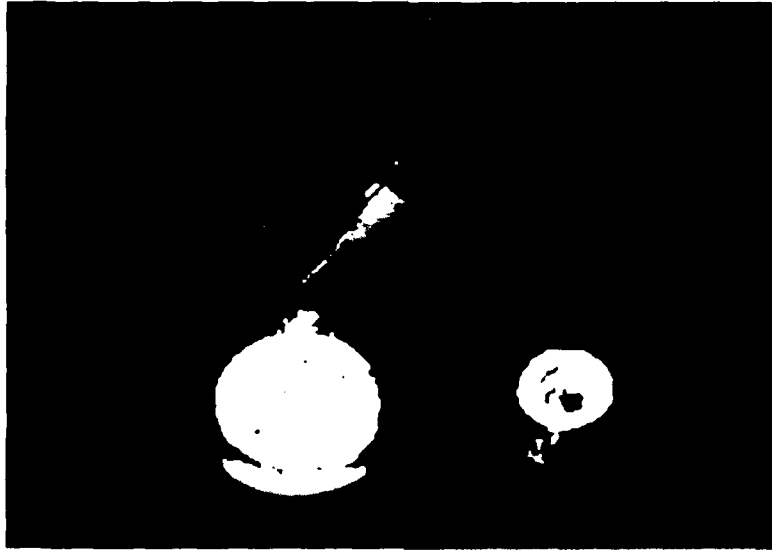


FIGURE 7 Z IMAGE WITH INCORRECT DATA

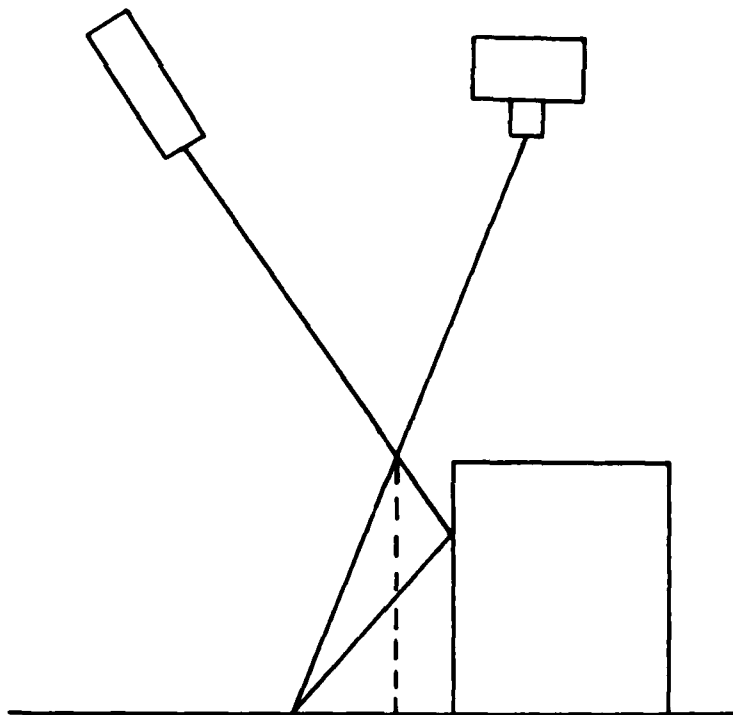


FIGURE 8 DIAGRAM OF A SECONDARY REFLECTION

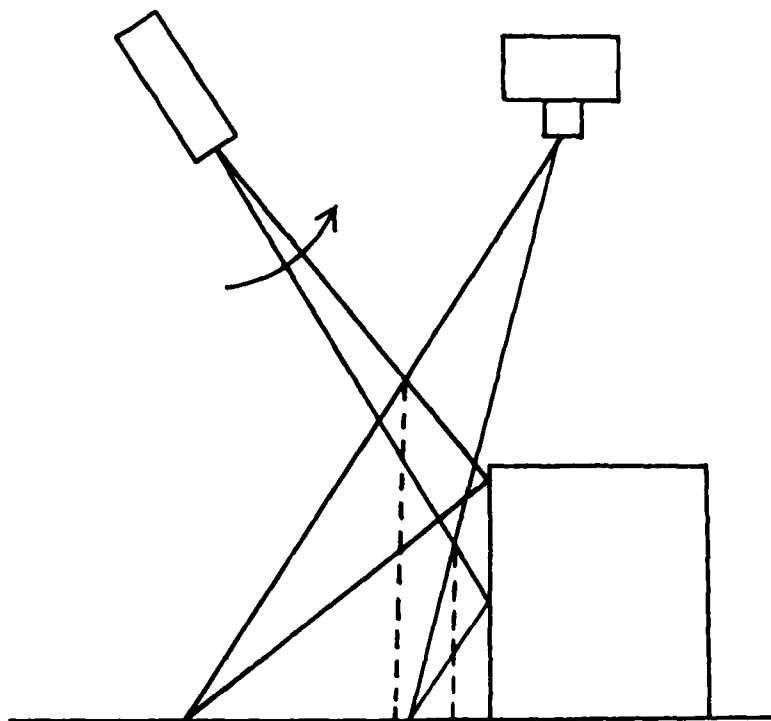


FIGURE 9 A SEQUENTIAL PAIR OF SECONDARY REFLECTIONS

the lower right. Notice that this filtering process missed some points near the bottom of the cylinder because data just below them were missing. Also notice that it correctly located small secondary reflections on the leading edges of some of the other objects.

One of the special cases in which the x coordinates can decrease as the light sheet is scanned along the x axis is when the object is transparent. Another case is when a thin object is suspended over another object and the camera and projector can see "around" the thin object. Figure 11 illustrates this case (which is known as an image reversal in the stereo-image understanding community). The x coordinate increases as the light sheet scans to the right until it hits the suspended object, then it takes a step backwards. The filter that we've described would mark this good data point as bad. Fortunately, this case does not arise often.



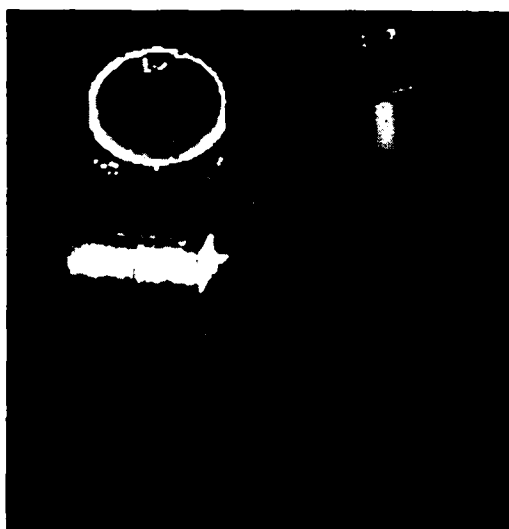
(a) RAW Z IMAGE



(b) X IMAGE



(c) POINTS WITH DECREASING
X COORDINATES



(d) CORRECTED IMAGE

FIGURE 10 FILTERING OUT SECONDARY REFLECTIONS

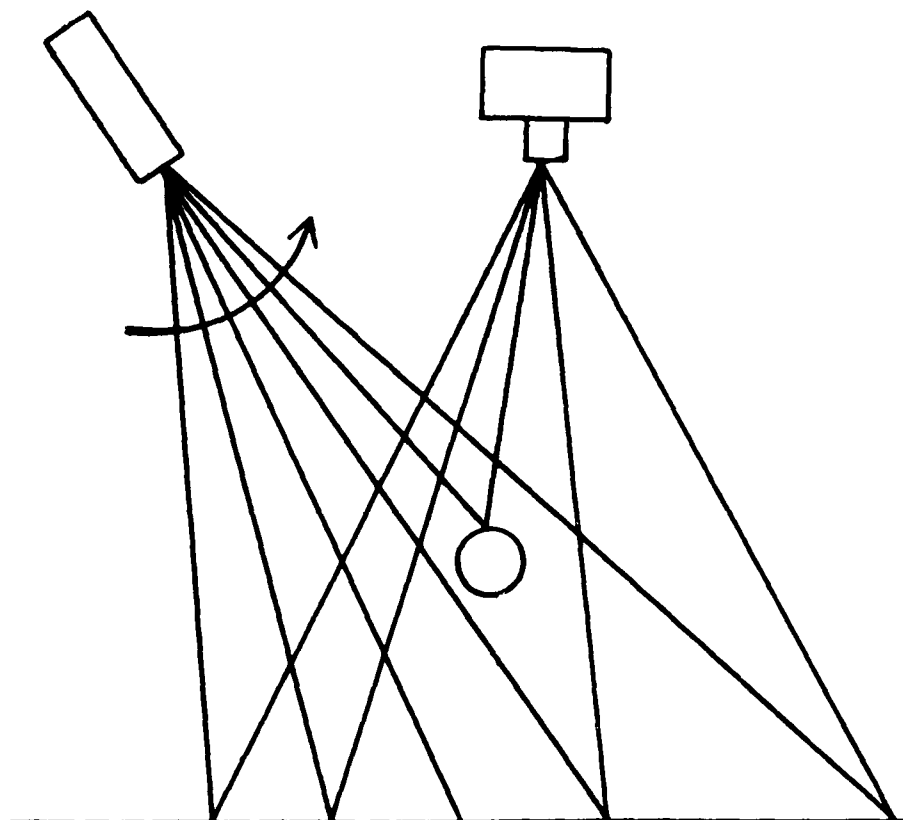


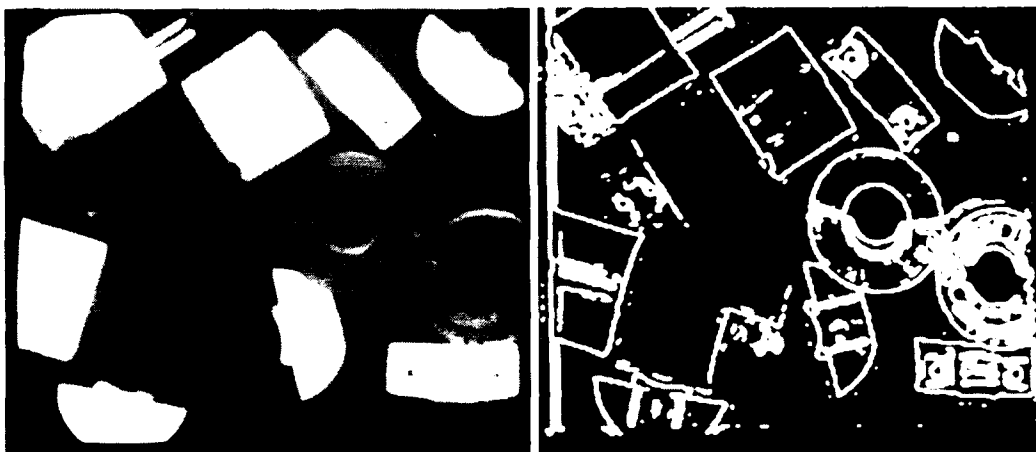
FIGURE 11 AN EXAMPLE OF AN IMAGE REVERSAL

IV CHARACTERIZATION OF FEATURE DETECTORS *

Automatic programming is a key to the success of flexible automation systems. Without it, a person will have to write a special-purpose program for each new task, and that is not economically feasible. Automatic programming for visual feedback, which is a cornerstone of flexible automation, consists of selecting a sequence of low-level processing techniques to detect features and a set of matching techniques to recognize objects. Figure 12 illustrates a typical sequence of processing steps. Given the gray-scale image in the upper left corner, the first step is to locate and link together edges (shown in upper right corner). The next step is to segment these edges into features, such as straight lines and circular arcs (shown in the lower left corner). And finally, distinctive clusters of features, such as pairs of parallel lines, are located and used to hypothesize possible part locations (shown in the lower right corner).

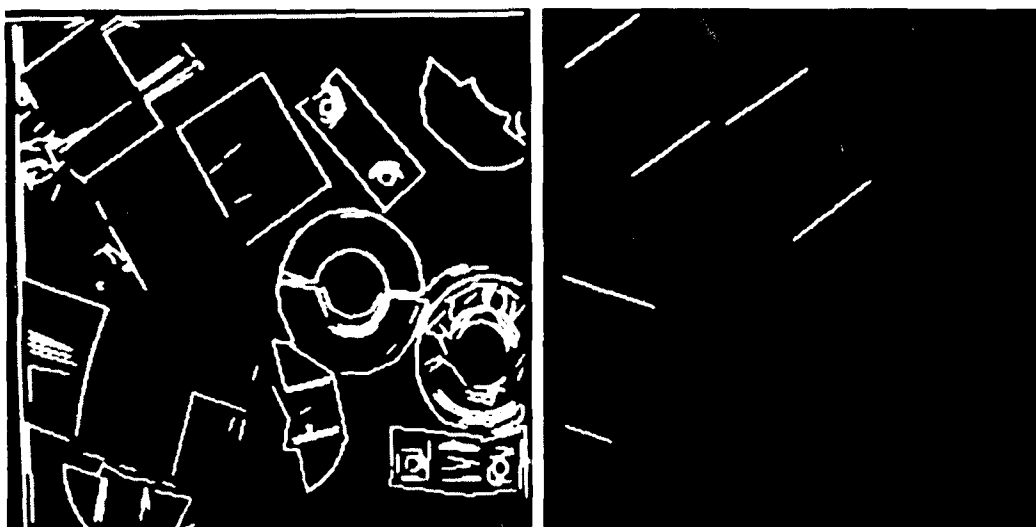
In order to select feature detection and matching techniques, an automatic programming system must know what the choices are, when they can be applied, and what results they are likely to produce. That is, the system needs characterizations of the techniques in a form it can use to list the possible ways of achieving a goal and to compare the possibilities according to their cost and reliability. These characterizations can be in the form of rules of thumb, or more preferably, analytic equations. For example, a rule might suggest the use of a "large" edge operator when the images are expected to be noisy, or an equation could explicitly state the minimum size of an edge operator to handle a specific amount of noise. Equations are better, assuming they are correct, because they are easier to apply. Unfortunately, they are also harder to formulate, because most parameters are dependent on a large number of interrelated factors.

* By Robert C. Bolles and James H. Herson



(a) RAW GRAYSCALE IMAGE

(b) DETECTED EDGES



(c) STRAIGHT LINES AND CIRCULAR ARCS FITTED TO THE EDGES

(d) PAIRS OF PARALLEL LINES MATCHING THE SIDES OF THE OBJECT

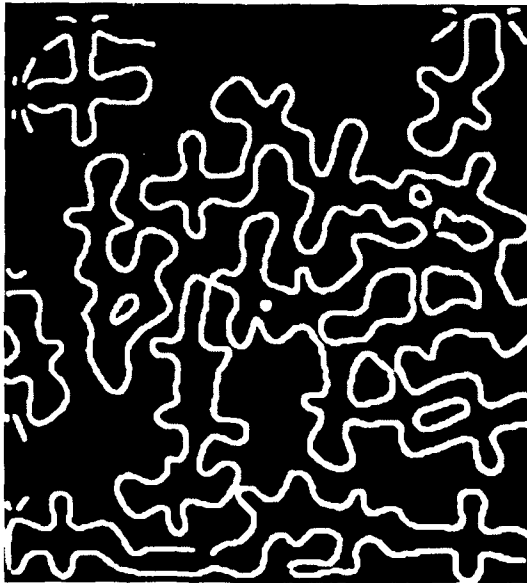
FIGURE 12 TYPICAL STEPS FOR RECOGNIZING AN OBJECT

One of our goals within this project was to select, implement, and characterize a set of techniques. We intentionally selected a variety of techniques, including some edge operators and some feature detectors. Most of the techniques were well known, although we have made a few improvements. In this report we describe our characterizations of a zero-crossing edge detector, which was originally developed by Marr and Hildreth at MIT [13], and a modified version of a split-and-merge line- and arc-fitting technique, which was originally developed by Pavlidis at Princeton [18, 19].

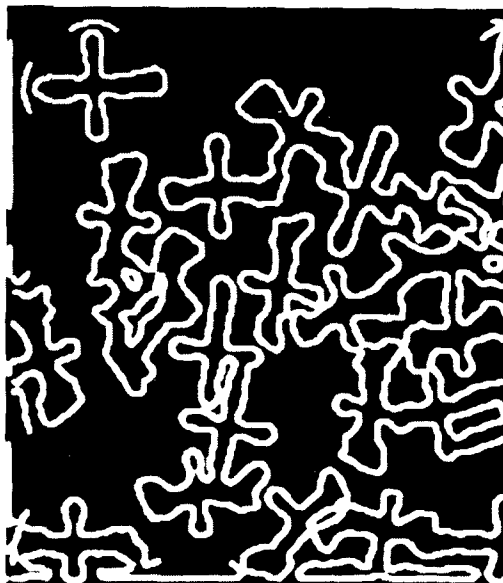
A. Zero Crossings

We implemented the Marr-Hildreth zero-crossing operator as a difference of Gaussians, which approximates the Laplacian of a Gaussian. This is a common approximation because the Gaussians can be computed more efficiently. The Gaussian is separable in the sense that the two-dimensional convolution can be done as a sequence of two one-dimensional convolutions, which significantly reduces the computation. Even with this simplification the processing time on a conventional computer is quite long (minutes on a VAX-11/750 for a 512 x 512 image). However, the operation has been performed efficiently on pipe-line graphics-type hardware, such as the Vicom, and on special-purpose hardware built at MIT. Because each pixel is processed identically, the computation of the two Gaussians, the differencing, and the thresholding could be nicely implemented in hardware. Some ingenuity will be required to produce an efficient "parallel" technique for linking the individual edge points into edges.

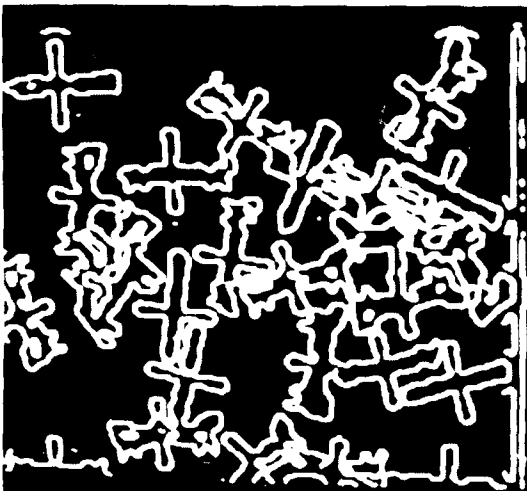
This technique has three parameters, the sizes of the two convolutions and the threshold for "significance." Determining the size of the smallest convolution is straightforward because it is a function of the noise in the image and the size of the smallest object features in the image. The kernel of the convolution has to be large enough to smooth out the noise and small enough to maintain the integrity of the smallest features. Figure 13 shows edges detected with kernels having standard deviations of 1 pixel, 1.6 pixels, 2.6 pixels, and 4.2 pixels.



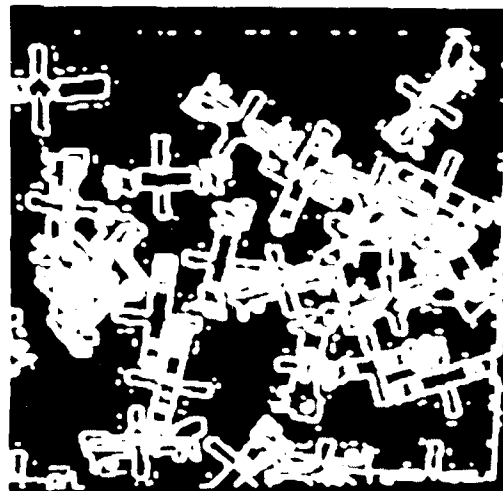
(a) 6x6 KERNEL



(b) 10x10 KERNEL



(c) 16x16 KERNEL



(d) 26x26 KERNEL

FIGURE 13 EDGES DETECTED WITH DIFFERENT-SIZED KERNELS

In their paper describing the zero-crossing operator, Marr and Hildreth argue for the second convolution to be 1.6 times the first. Their argument was based on bandwidth considerations. Other people have argued for 2 times the first size on the basis of computational efficiency. We have followed the suggestion of Marr and Hildreth.

The third parameter is somewhat more difficult to set. We set it by analyzing the graph of the number of edges accepted at different thresholds. Figure 14 shows the edges accepted at four threshold values. The graph generally takes the form of a decaying exponential (see Figure 15). We pick a threshold slightly to the right of the knee of the function.

The two most positive properties of the zero-crossing edge detector are that it produces thinned, connected edges and that it performs a type of lateral inhibition. The process of locating sign changes in z on a three-dimensional surface that fluctuates back and forth across the $z = 0$ plane inherently produces closed contour-like curves that are a single pixel wide. This property eliminates the thinning processes and the gap-filling heuristics, which are required by conventional edge detectors, such as the Sobel operator.

The process of lateral inhibition subtracts the local average from each point in an image in order to correct for DC changes in amplitude or smooth gradients. Its use in an edge detector focuses attention on relative changes in brightness rather than absolute changes.

The negative attributes of a zero-crossing edge detector are that it rounds off sharp corners; it is forced to select one path through a T junction; and it introduces spurious edges. Figure 16 shows edges obtained with a large kernel size. The rounding is due to the smoothing part of the computation, which blurs the original image. Blurring a straight edge does not change the zero crossing's estimate of the edge's location. However, blurring a sharp corner rounds the edge by a factor that is a function of the size of the convolution and the size of the angle. A good rule of thumb is that the largest deviation for acute angles is approximately equal to the standard deviation of the smallest kernel.



FIGURE 14 EDGES PASSING DIFFERENT MAGNITUDE THRESHOLDS

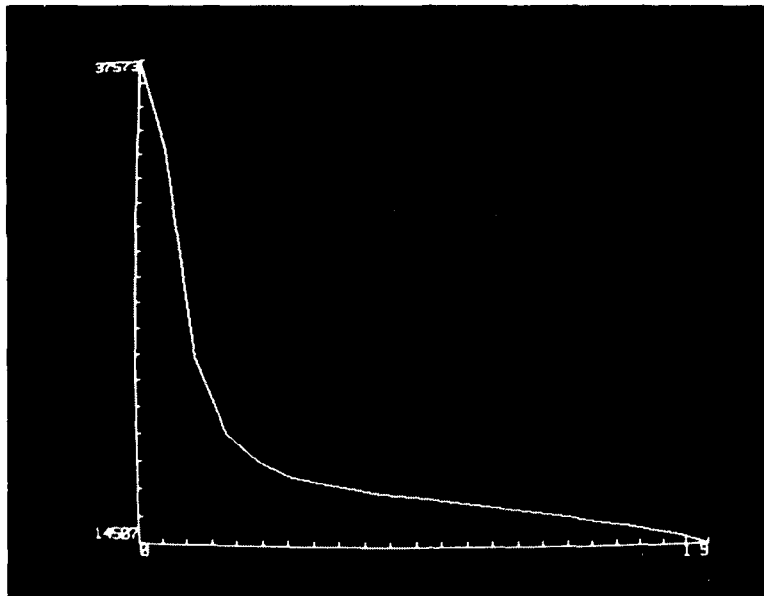


FIGURE 15 GRAPH OF THE NUMBER OF ACCEPTED EDGES AS A FUNCTION OF THE MAGNITUDE THRESHOLD

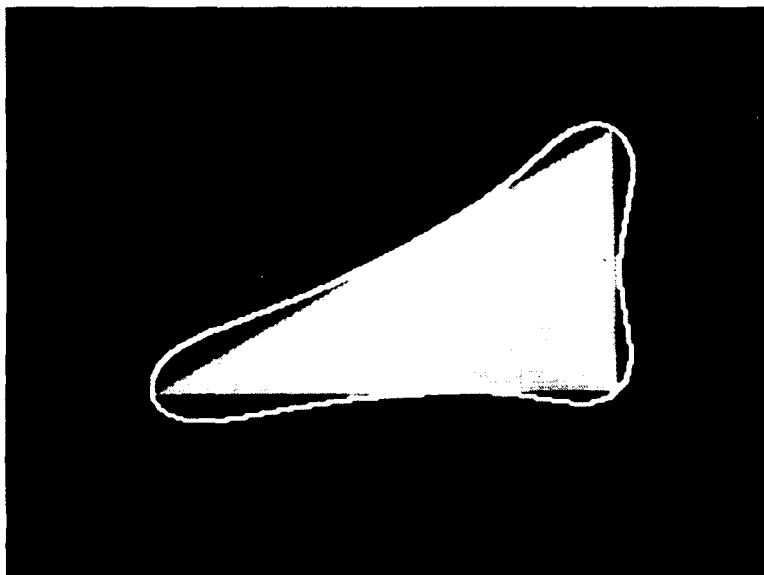


FIGURE 16 AN EXAMPLE OF THE DISPLACEMENT OF EDGES CAUSED BY LARGE KERNEL SIZES

Another manifestation of the smoothing performed by the zero-crossing operator is that an isolated large noise spike will produce a circular edge whose radius is approximately equal to the standard deviation of the smallest kernel. This occurs because the smoothing operation rounds off the spike into a gentle hill around which a zero-crossing appears.

A T junction in an image most often occurs when one surface occludes a pair of other surfaces. Thus, there are three regions in the image that meet at a point. Because zero crossings are forced to form contours (i.e., closed edges that don't cross each other), the edge at a T junction is forced to divide the area into two regions. Not only does this process miss one of the regions, it also misses the fact that there is a T junction, which may provide important clues about occlusion in the scene.

Another result of the necessity for the zero crossings to be connected curves is that they may be forced to go through a region of the image that does not have any obvious edges. When this happens, the edges often resemble space-filling curves. They wander like a lazy river all over the region. Heuristics for eliminating these spurious edges are based on their significance values, mentioned above, and their high-frequency curvature changes.

In summary, a zero-crossing edge detector has its strengths and weaknesses, as do all edge detectors, and the choice of which one to use is a function of the task. A wise choice can be made only by a person or automatic programming system that has a description of the task and descriptions of the strengths and weaknesses of the alternatives.

B. Line and Arc Fitting

We implemented a split-and-merge segmentation technique that segments a two-dimensional edge into circular arcs and straight line segments. The technique makes an initial segmentation and then tries to merge adjacent segments into larger arcs or lines. The initial segmentation recursively partitions the edge into smaller and smaller segments. It tries both lines and arcs. If neither fits well, the segment is partitioned into two segments at the point farthest from the line joining the end points. If one of the models fits, it is accepted. If both fit, the user must specify a preference for arcs or lines. It is relatively difficult to set up a criteria to compare a fitted line with a fitted circle; an arc can always fit better, because it has one additional degree of freedom. We, and others, have considered several techniques, such as requiring smaller errors for arcs, but we are not particularly happy with any of them. One approach we've taken is to allow multiple explanations for parts of the curve, which delays a choice until the feature-matching step. The cost is an increase in the number of features to be considered.

This technique has four parameters: the error threshold for fitting lines, the threshold for arcs, the minimum segment length, and a preference for lines or arcs. The fitting thresholds are functions of the quantization error and noise; the segment length is a function of the smallest features to be detected. And the preference for lines or arcs is a function of the recognition strategy. Figure 17 illustrates the results produced by applying this technique with three different parameter settings. Figure 18 shows its results on a more complicated set of curves.

The computational requirements of this technique are high, because it requires several passes through the data. The line- and arc-fitting routines require a pass through the list of points, and the analysis of the errors requires another pass through the points. The total number of passes through the whole list of points is approximately log to the base 2 of the number of segments produced by the initial recursive segmentation step, which is a function of the number of different segments along the curve.

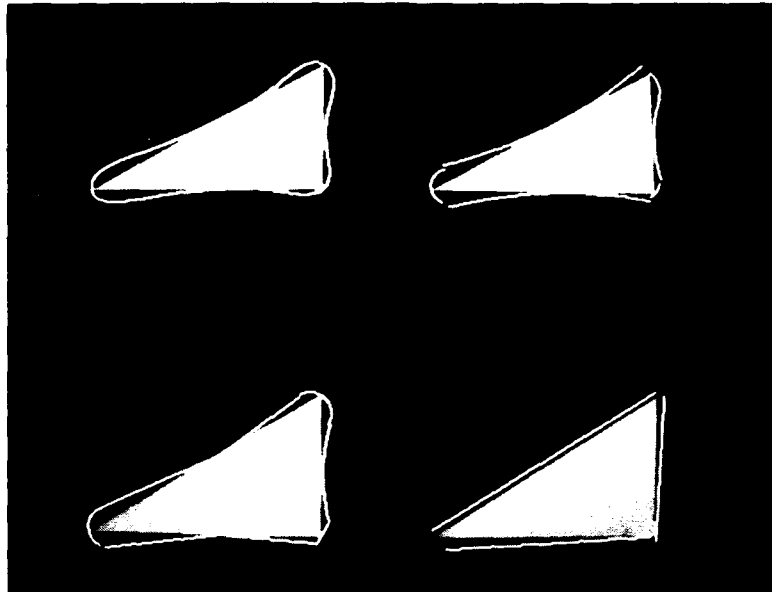


FIGURE 17 ARC AND LINE FITTING WITH DIFFERENT PARAMETER SETTINGS

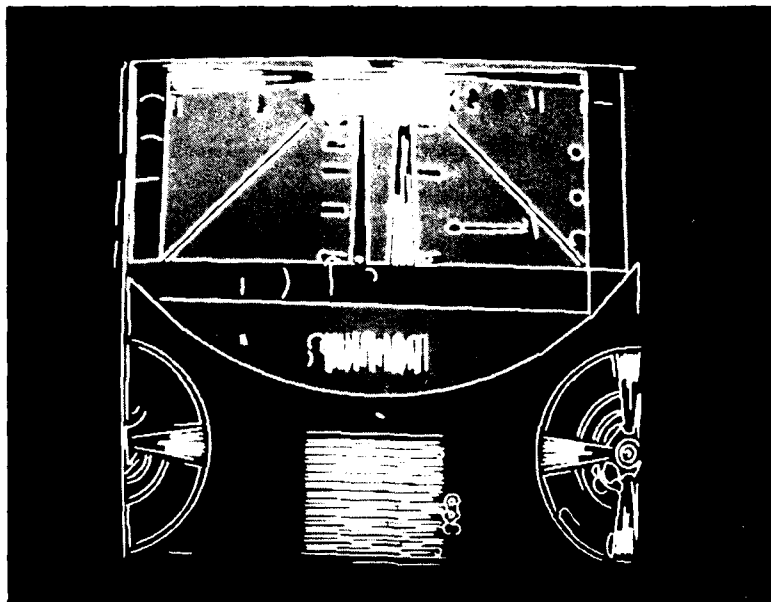


FIGURE 18 ANOTHER EXAMPLE OF ARC AND LINE FITTING

The positive attributes of this technique are that it fits both lines and circular arcs to the edges and that it tries to improve its segmentation by merging adjacent segments. However, because it is forced to select one explanation for each part of an edge, it sometimes makes a mistake on ambiguous configurations. We often apply the procedure twice, once preferring lines and once preferring arcs. Any portion of the curve that is labeled the same way by the two applications is assumed to be correctly labeled. The others are left ambiguous.

Another problem with this technique is that it has trouble locating a junction between two curves that join smoothly, such as a line that joins an arc so that the line is approximately tangent to the circle at the junction. Because the transition from one segment to another is unclear, the location of the transition is somewhat random, depending on the position and order of the break points inserted by the recursive segmentation.

V MULTIARM COLLISION AVOIDANCE *

A. What Is Collision Avoidance?

The problem known as "collision avoidance" is concerned with how to specify a path from the current position of a robot manipulator to a specified goal, without having the robot collide with surrounding objects. The collision-avoidance system must read the current location of the robot and construct an internal model of the situation. It then must plan a path that gets the robot safely to the goal position. The path is generated and verified by interacting with a collision-detection module that can tell when objects are in intersection with the arm.

Multiarm collision avoidance is concerned with performing the same task with more than one robot, when the robots are in motion simultaneously. Not only must each arm be simultaneously concerned about avoiding surrounding obstacles--the arms must also avoid each other. This would not be so difficult if one arm were moving while other arms kept still. However, that would be inefficient. All arms must be permitted to move simultaneously. Thus, time-dependent obstacles are introduced into the problem, making a representation of time essential in a simulated system. To the author's knowledge, SRI is the only research center that has explored the difficult problem of collision avoidance with multiple articulated robotic arms.

* By John K. Myers

B. The Difference Between Collision Detection and Collision Avoidance

A predictive collision-detection system uses simulation to answer the relatively straightforward question of whether a given movement will result in collision or not. If the movement does not present difficulties, then the (actual) robot may be permitted to execute that move. If, however, the movement results in a collision, a collision-detection package can only inform the user about that event.

A collision-avoidance system, in contrast, addresses the much more difficult problem of how to plan a path around obstacles once the original path is known to be blocked. Typically, a collision-avoidance system will use a collision-detection package to test out different proposed movements, or just to make sure that nothing has gone wrong.

C. The Motivation for Collision Avoidance

Solving the collision-avoidance problem is important because it raises the level of programming of the robot system. The job of a robot programmer might be to design and implement a new assembly sequence for a robot. While doing this, much of his time is spent in finding appropriate approach positions, withdrawal positions, and intermediate poses that will permit the robot to do its task without colliding with surrounding objects. A collision-avoidance system eliminates the time wasted on this effort. It allows the programmer to specify the sequence in terms of what the robot is supposed to accomplish, and removes the concern about the low-level details of the movements.

Similarly, a collision-avoidance package is essential for automatic assembly systems. Researchers at SRI are beginning to attack the problem of having a computer program plan the assembly of a product. The automatic assembly-planner program must have confidence that when it specifies transfer motions required by the plan, the robot will be able to carry out these motions. The assembly planner is concerned with high-level details about the assembly sequence. It requires, as its complement, a collision-avoidance system to be concerned with the low-level movements.

A collision-avoidance system is also required in situations involving locational uncertainty and sensory feedback. For example, perhaps a part is known to be on a table somewhere between several large objects, but its exact position is unknown. A camera is used during run-time to report the part's position. The robot is then required to pick up the part. In such situations, when the sensory feedback requests robot motion in a particular direction, that motion must be verified with the collision-avoidance system. Otherwise, a direct motion of the robot to the part could result in an unanticipated collision with surrounding objects. Previous efforts have relied on operator programming to account for all possible input and ensure collision-free behavior.

D. Other Approaches

Collision avoidance with even a single rotary-jointed arm in three dimensions has been generally conceded to be an extremely difficult problem [5]. Most other approaches to collision avoidance have begun by making various simplifying restrictions.

Brooks only allows objects to be carried flat, without any horizontal rotations [5]. He also requires objects to either stick straight up from the floor or down from the ceiling, thus precluding overhanging obstacles. This allows him to reduce the problem to a small number of two-dimensional path-planning problems, which may then be solved relatively easily.

Khatib pioneered the use of the potential-field approach (see [8, 7]). He demonstrates a real-time (single-arm) system with six geometric primitives in its environment. For speed, he models the arm using three capped cylinders shrunk to line segments. Held objects must be modeled by an additional segment, using application-specific routines.

The author's previous approach in [14] and [15] uses a heuristic based on avoiding "entire objects" to search alternative paths to the goal until a successful one is found. The approach can accept overhanging obstacles, large objects held in the hand, and moved objects. However, it does not generate "smooth" paths, and sometimes takes more than 30 seconds to search for a collision-free path.

Most importantly, it does not generalize easily to simultaneous motions of more than one manipulator.

Other approaches, e.g. Lozano-Perez and Wesley [12], Udupa [24], and Lewis [10], have operated on even more restricting assumptions.

E. Accomplishments

Under the AFOSR contract, SRI has have developed a system that performs general multiarm collision avoidance in near real time, using the potential field approach. The developed system is built around the existing Robotic Simulator with COLLISION DETECTION SYSTEM (RCODE) [15, 16], which was implemented previously. We have investigated alternative methods of implementing a potential field collision-avoidance system, with associated tradeoffs between computational effort and accuracy. We have implemented a unique collision-avoidance system that considers the entire volume of the arm, operates with multiple arms simultaneously, allows arms to pick up objects and performs avoidance with the new configuration of the arm holding the object, and is capable of operating using detailed geometric models with many primitives. Finally, we have implemented and demonstrated this system on an actual assembly station with two robot arms and numerous surrounding obstacles.

F. Description

Collision avoidance research efforts focus around the SRI Assembly Test-Bed Station. Currently, this station consists of two Unimation 560 PUMA robots, each mounted on its own table; a table for parts; a three-degree-of-freedom x-y- θ table; and two cameras in the ceiling, which is low enough so that a robot could hit the cameras. The end-effector of one arm consists of a wrist force sensor and a gripper and that of the other arm consists of an RCC, a gripper, and a TV camera front end. A large cardboard box is placed on the table in front of one of the robots, as a test object.

Each of these objects is described in a geometric model used by the system (see Figure 19). Each robot is composed of about 35 geometric volume primitives. The x-y- θ table, together with the rest of the surrounding objects, are modeled by

an additional 30 primitives. The algorithms developed under this project enable the system to compute paths using these models in about real-time. No other research effort has used such a detailed model to perform collision avoidance.

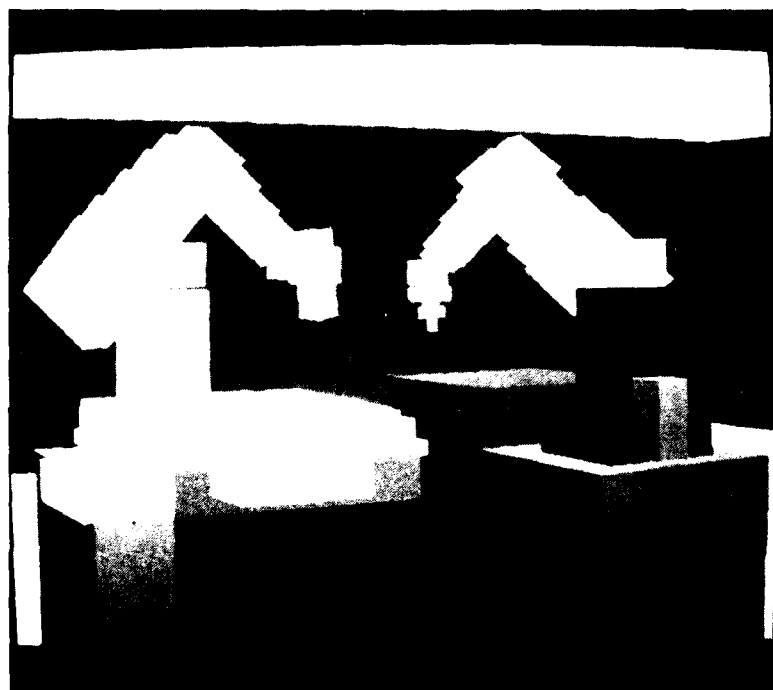


FIGURE 19 MODEL OF THE ASSEMBLY STATION

G. The Potential-Field Approach

The potential-field approach to collision avoidance is based on the concept that the goal generates attracting forces while obstacles generate repelling forces. The arm's motion is basically a function of its current position only. The robot is moved using local information--that is, just considering obstacles that are immediately close to the arm. A mathematical potential field, analogous to an electrical charge field, is generated around the models of the objects. This field acts to repel the arm. An attraction is placed at the goal position which acts to draw the arm towards the goal. A single pose of the arm is considered. The influences operating on the arm are evaluated, and the resulting force is used to

move the arm a small distance. The process is repeated until the arm attains its goal, or until the function decides that it has failed.

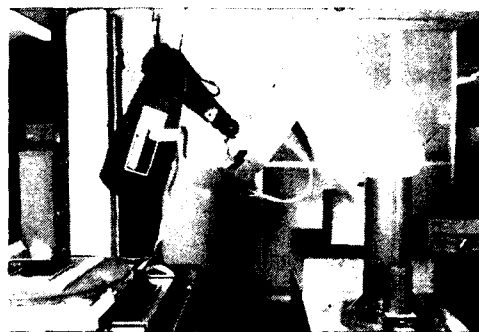
Part of the problem in using the potential-field approach to collision avoidance involves specifying which of a number of alternative algorithms to use in computing the arms' movement. These include how to specify the attraction towards the goal, the field around the obstacles, the combination of the forces on each arm, and the resulting incremental motion of each arm model. One of the main contributions of this study is in exploring the existence of these alternatives (see Appendix D).

H. An Example

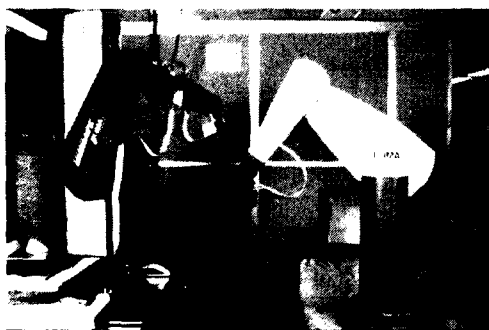
In Figure 20, the illustrations, we see two robots which have been ordered to move to different locations (Figure 20a). The black robot, on the left, has been ordered to move to a location in the foreground; the white robot, on the right, has been ordered to move to a location in the background. If they were to move normally, without using collision avoidance, they would run into each other in the middle of their paths. However, the collision avoidance system is able to simulate their motion and compute safe paths for both of them. The simulated arms "feel" each other coming as they get closer (Figure 20b); the black arm is pushed up, and the white arm is pushed down slightly, until they can safely get past each other (Figure 20c). They then reach their respective goals successfully (Figure 20d). This example illustrates that the system is capable of controlling two arms simultaneously, even when the arms are performing complicated motions with many rotations.



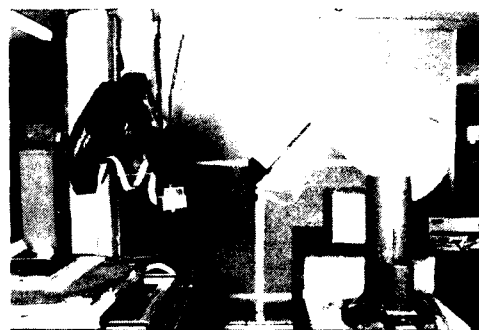
(a) ROBOTS BEFORE
ATTEMPTED MOTION



(b) ROBOTS "FEEL" ONE
ANOTHER COMING



(c) ARMS MOVE TO ALLOW
SAFE PASSAGE



(d) ROBOTS REACH GOALS
SUCCESSFULLY

FIGURE 20 SIMULTANEOUS COLLISION AVOIDANCE WITH TWO ROBOT ARMS

I. Advantages and Disadvantages of the System

The collision-avoidance system developed under this project is capable of performing collision avoidance for many robots moving simultaneously in the same workspace. It is demonstrated with two PUMA robots. Collisions are avoided not only between each robot and the surrounding environment, but between pairs of robots as well. The program is capable of handling robots with any combination of translational or rotational joints, and with more or less than six degrees of freedom. It could be extended to situations involving asynchronous robots or other moving machinery not controlled by the collision avoidance module.

The system uses a detailed model of the situation, with many geometric primitives. Complex workstations can be modeled faithfully and cost-effectively. Because of a unique hierarchical approach to spatial modeling, low-level details of complex objects can be filled out and modeled, without the normal corresponding slowdown in the system due to increased complexity.

The developed system is comprised of different modules that can be configured to trade off between speed and accuracy (how "appropriate" or "esthetic" the solution is). If the requirements are for a fast solution that just gets the arm to the goal, that can be done; if the user desires a "good" path, and is not so concerned about how long the system takes to derive it, that can be done as well.

The system uses the potential field approach, a local method, to search for a path to a specified goal. It quickly generates smooth paths around objects that would result in glancing collisions. It is able to avoid certain kinds of collisions that pose most of the difficulties in a typical transfer task. The system does this easily, without using complex calculations as most other methods are forced to do.

Most other systems start by approximating the hand by an enclosing sphere, and then allowing only small objects to be grasped. Our collision-avoidance system can handle the case of allowing the robot to arbitrarily pick up a large object. Collision avoidance is then performed with the entire volume of both the

robot and the object, with respect to other robots and the environment. No previous special computations are required to change the system; no restrictions are made as to where the robot is permitted to grasp the object.

The main disadvantage is that the potential-field approach does not solve the collision-avoidance problem completely. There is no guarantee that the arm will actually ever reach the goal, because the arm can be trapped in potential wells, and because certain other types of collisions become very complicated and difficult for the method to solve. For instance, a head-on collision with the side of a large box is difficult because the robot doesn't know which way to go. The force away from the obstacle cancels the force toward the goal, leaving the robot with practically no force component for motions. This can occur even when there are no actual potential wells. The approach lacks a global perspective that would permit it to understand obstructing situations, and allow it to go entirely around blocking objects in one motion.

One of the other disadvantages of the system is that it is still too slow. Although it searches only a single path for each robot, without using backtracking, it achieves a rate of about 3 to 10 Hz, which is well below arm servo rates. This is primarily due to the complexity and detail of the models involved, and the fact that the entire volume of the arm, not just a three-segment approximation, is used in calculations. A faster computer or special coding could of course be used to increase this speed.

J. Remaining Areas of Research

A complete solution to the collision-avoidance problem has yet to be found. Although the current system uses the potential-field approach to obtain tolerable solutions for most problems, it still cannot be said to truly "understand" the situation. There is a need for a collision-avoidance system with intelligence, that can take a holistic view of the situation. For instance, if there are three obstacles on a table, the robot should avoid all of them at the same time--not run into the second while going around the first. Not only that, an ideal system would be able to remember its mistakes, and use them to avoid redundant testing in current situations. Such high-level knowledge is difficult to represent.

Collision-avoidance systems must be general enough to work well in complex environments and in real time with any kind of manipulator. Further original work is necessary to extend the state of the art, as current systems are fundamentally constrained by one or more of these requirements. For instance, the robot must be able to find its way around any type of obstacle, including head-on and overhanging obstructions: Situations requiring paths around asynchronous second manipulators or machinery not under the control of the path-planning computer still present problems.

There remains the serious difficulty of how to plan optimal paths when the time spent computing is not a problem. It will probably be difficult to prove that a path is optimal. In this case, a program that can find "almost optimal" or "desired" paths would be useful, instead of the current system which just locates some path that works. Criteria for desirability vary, and might include paths that are shortest (in cartesian or joint space), fastest, straightest, or that expend the least energy. A problem related to path optimality is being able to prove when a desired movement has no solution. The current system can only report that it has failed to find some path, and cannot prove no path exists.

Unsolved problems related to collision avoidance include coping with moving objects that exert certain contact forces on each other, such as sliding motions, insertions, and extractions; planning small, intricate motions with little maneuvering room; and planning how to grasp and place objects properly so that the objects don't fall out of the hand and the areas of interest are free to be worked with (commonly known as "grasp planning").

In summary, the current system is not capable of solving all problems presented to it, offers no guarantee of optimality, and possesses a limited "understanding" as to the situation presented to it and the appropriateness of its actions. Further progress is required in all these areas.

VI COORDINATION OF MULTIPLE MANIPULATORS

A. Introduction

We propose a method for increasing the speed with which a computer can reason about spatial relationships. As more and more robots are introduced into industry, this will become important first at the work station level, then at the production line level, and later at the factory level of automation.

B. Motivation

Software to operate multiple, cooperating robots efficiently and safely will be too complicated for human programmers to write quickly enough or (more importantly) correctly enough. An automatic factory will therefore have to program itself, much of the time, and in parallel with execution of the previous job, for high productivity. The factory will be even more productive if it can analyze manufacturing failures when they occur and revise its original manufacturing plan dynamically to overcome them. Fast algorithms for spatial reasoning will be important both for self-programming and for run-time failure correction.

C. Method of Approach

Our method is to precompute a kind of "map" showing all the robot positions that will result in a collision. Although generating this map is a slow process, consulting it is rapid enough for real-time collision avoidance during production.

The collision map of a workstation tooling arrangement for a particular batch production run can be generated automatically by computer while the previous batch is being produced. Computation time is therefore not a serious constraint, but we have nonetheless found a way to partition the problem and reduce the time requirements. The amount of data in a typical map is quite large,

but we found that it is highly redundant, so a simple storage compression technique can greatly reduce the computer memory needed to store a collision map.

We generated collision maps for two simple robot arms in the presence of some obstacles and used them to coordinate the robots' motions in a simulation. We also developed techniques for using the collision map to solve some other common robot control problems, but did not test them.

1. *Collision Map in State Space*

A collision map shows where collisions occur in the multidimensional state space of robot joint positions ("joint space"). It is represented in the computer by a binary-valued, multiply subscripted array A . For any state of the robot system given by joint positions J_1, \dots, J_N , $A[J_1, \dots, J_N] = 1$ means a collision occurs, 0 means it does not.

We compute the value of A for all combinations of joint positions with a resolution of, typically, 32 different positions for each joint over its range of motion. For each set of joint positions, we check for collisions using a rapid three-dimensional geometric modeling program (written by John K. Myers [14]) to simulate the robot arms. We call this "surveying the state space."

For example, Figure 21 shows the two simple two-jointed robot arms that we simulated in our experiments. The ceiling, floor, and two boxes B_1 and B_2 are obstacles to avoid. The left arm has two rotary joints whose positions are angles J_1 and J_2 . The right arm has a rotary shoulder with angle J_3 and a boom that extends a distance J_4 .

Figure 22 shows some representative two-dimensional cross sections through the four-dimensional state space of this robot system. In each cross-section, two of the joint positions are held constant while the other two vary across the diagram, as indicated in the legends at the top. The value of A is 1 in the dark "obstacle regions," indicating configurations in which one of the arms collides with itself, the floor, the ceiling, one of the boxes, or the other arm.

2. *Collision Map Compression*

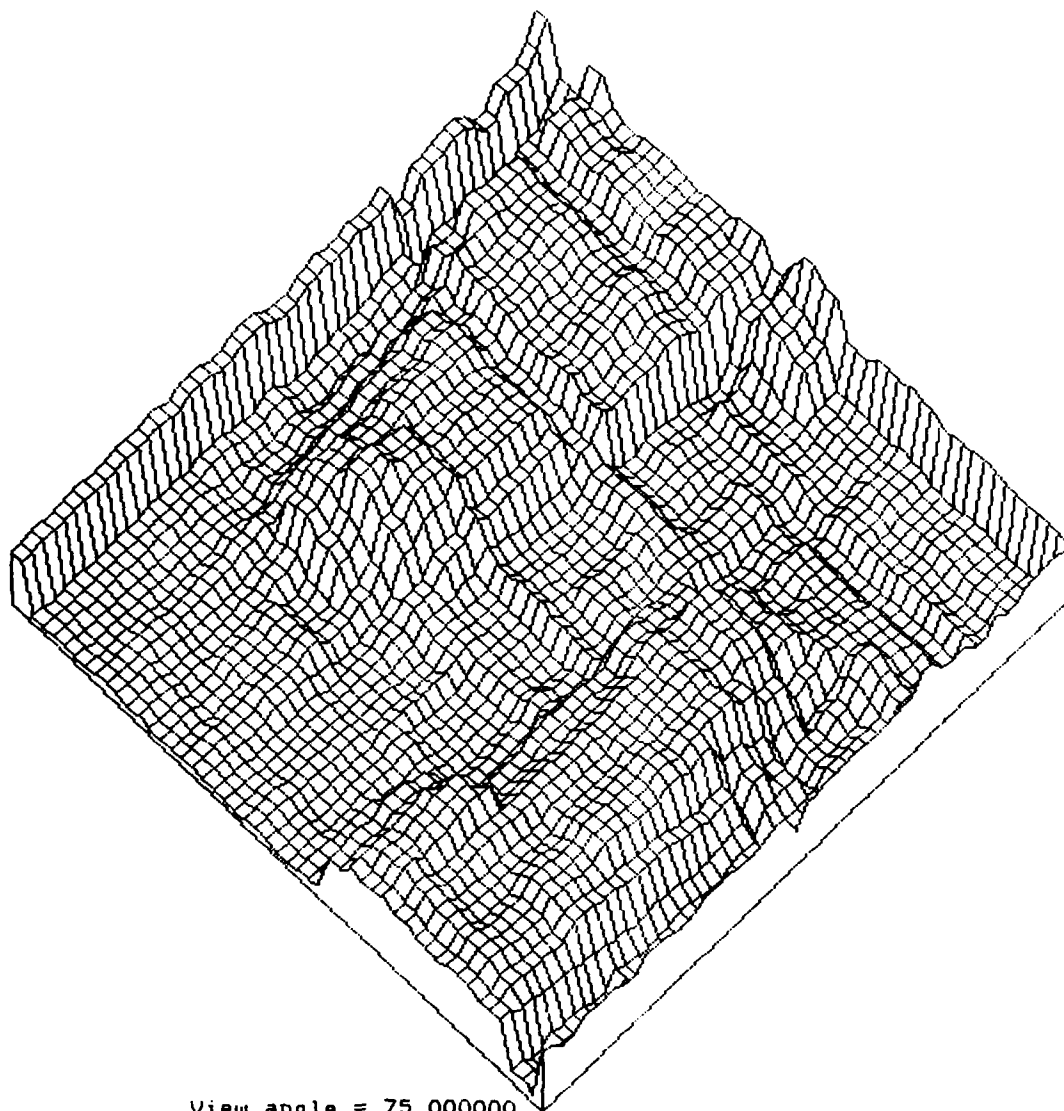
Collision maps are large, but because the information in them is very repetitive, they can be compressed to save computer memory. We experimented with an unusual storage compression technique--Albus' Cerebellar Model Arithmetic Computer (CMAC) algorithm [1]--because it had a useful side effect. It smoothed out the contents of a collision map, converting each binary "collision/no collision" value to a number between 0 and 100, indicating nearness to an obstacle. This allowed us to implement a simple collision-avoidance algorithm that used only information in the map local to the point representing current state of the robot system.

Figure 23 shows a typical plot of nearness values over a cross section of a collision map after compression and smoothing by the CMAC algorithm. The surface is high where collisions occur and low in safe regions. Using the negative gradient of this function to indicate the "downhill" direction, the global collision-avoidance rule, "avoid the obstacle regions," reduces to the simpler local rule, "stay in the valley." We experimented with this control method in the robot simulation with generally good results, although the "ripples" visible in Figure 23 (probably artifacts of our implementation) sometimes caused problems.

3. *Uses for a Collision Map*

Some different ways to use a collision map include:

- **Collision Checking**--Will a collision occur for a given pair of arm postures, say $(J1=N1, J2=N2)$ and $(J3=N3, J4=N4)$? **Method:** Read out the value stored in the state space table at the point $(N1, N2, N3, N4)$. If the value is 1, a collision will occur; otherwise, it is safe.
- **Trajectory Planning**--How can we move the left arm from posture $(J1=N1, J2=N2)$ to $(J1=N3, J2=N4)$ when the right arm is in posture $(J3=M1, J4=M2)$, without hitting anything? **Method:** Look for a path from point $(N1, N2, M1, M2)$ to point $(N3, N4, M1, M2)$ that goes around the obstacle regions. If the path can stay in the $J1$ - $J2$ plane through these points, it will not be necessary to move the right arm out of the way.



View angle = 75.000000

Left front axis: # 2 -180.00 to 180.00
Right rear axis: # 3 0.10 to 1.50
Joint 0: 0.00
Joint 1: 0.00

FIGURE 23 COLLISION MAP CROSS SECTION AFTER COMPRESSION
AND SMOOTHING BY CMAC

- **Multiarm Coordination**--How can both arms move safely and simultaneously from $(J1=N1, J2=N2)$ and $(J3=N3, J4=N4)$ to $(J1=M1, J2=M2)$ and $(J3=M3, J4=M4)$? Method: Find any path from $(N1, N2, N3, N4)$ to $(M1, M2, M3, M4)$ around the obstacle regions.

4. *Comparison to Work by Others*

"Configuration space" is the term used in the literature for our "state space," while the problem of avoiding collisions is called the "findpath" problem. Moving a single rigid object around obstacles is the "piano mover's problem."

Early work on collision avoidance concentrated on single objects moving in the plane, which have three-dimensional state spaces. Howden [6] introduced the idea of quantizing the state space and searching it for collision-free paths. Udupa [24] effectively generated approximate obstacle maps in a three-dimensional state space for planar manipulators with two and three joints by "growing" the obstacles. Lozano-Perez represented obstacles by geometric figures [11] and first presented the Vgraph (viewability graph) algorithms for finding clear paths in high-dimensional state spaces [12]. Reif [20] proposed methods for representing state-space obstacles exactly with multinomial algebraic expressions, while Wallace [26] has achieved interesting results in transforming certain kinds of obstacle shapes into a multidimensional state space. Schwartz and Sharir [21] using topology theory were able to solve the findpath problem with rotations in two- and three-dimensions by means of polynomial-defined surfaces in a 12-dimensional state space. Brooks and Lozano-Perez [3] presented a recursive subdivision strategy for the findpath problem, while Brooks [4] used generalized cones to represent the (physical) space between obstacles.

Our method combines the idea of using a high-dimensional state space and quantizing it. However, instead of analytically transforming simple physical obstacle shapes into regions of state space, we propose a direct survey of many discrete points in the state space as described above. The growing power of specialized VLSI computer hardware, matched with the relatively long duration of batch manufacturing runs, makes this computationally intensive approach an

increasingly attractive alternative. For example, using a rapid three-dimensional geometric solid modeling system [15], in only seven hours on a conventional serial computer (a Digital Equipment Corporation VAX-11/750 with 2-megabyte memory), we were able to survey 1 million different positions of two two-joint robot arms in a space with four obstacles. Heuristics of various kinds could further speed up these computations. Finally, use of a general-purpose geometric modeling system will allow more accurate prediction of collisions between arbitrarily shaped (especially curved and concave) objects than is possible with the usual coarse, polygonal shape approximations.

Independently of the method used to generate the collision map, we propose that it should be stored explicitly in a compact format suitable for rapid lookup, rather than implicitly as, say, 12th-order polynomials. This will speed up run-time solution of findpath problems for error recovery.

We believe that this is the first application of these techniques to coordination of multiple manipulators. We also think it may be the first unified approach to a number of "awkward" manipulator-control problems, such as planning trajectories to avoid joint limits, kinematic singularities, and configuration changes.

Our local collision-avoidance method most resembles Khatib's [7] in that we also simulate a repelling field around the obstacles (the "downhill" component of gravity). However, our pseudo-forces act in the multidimensional state space, while Khatib's act in the three-dimensional physical space around the robots.

D. Future Work

Some of the many topics that need to be pursued to turn our research results into a practical robot-control system include:

- Development of improved methods for dealing with higher-dimensional state spaces (i.e., more robot arm joints).
- Specialized geometric modeling techniques or transformation methods, such as Wallace's [26], to speed up the state space survey.

- Investigation of recent advances in storage compression, such as multidimensional run-length encoding [25] and multitrees [27].
- Novel approaches to global trajectory planning, such as optimization by simulated annealing [9].
- Exploration of the possibilities engendered by introduction of a time axis in the state space.
- Study of specialized hardware implementations of some of the steps of the process based on advanced computer hardware technology, such as bubble memories, associative memories, laser discs, holographic data storage, and special-purpose VLSI circuits.

VII PROGRAMMABLE ASSEMBLY TEST-BED *

The final component of the AFOSR project is the extension of the SRI Programmable Assembly Test-Bed. The test-bed is an assembly station, which serves to integrate much of our work. The test-bed has a modular, distributed-processing architecture based on a local-area network (see Figure 24). Each module in the station consists of some major device (e.g., a manipulator), auxiliary devices (e.g., an end-effector), and its own computer to interface it to the network and perform local processing. A library of functions is defined for each module; a particular function is activated by sending a command over the network to the module. The distributed architecture enables concurrency in the station operations, previously demonstrated in the assembly of a subcomponent of a computer printer [22].

As part of the AFOSR project, a number of extensions were made to the station:

- A VAX-11/730 was acquired (at no cost to the sponsor) and made the real-time coordinator of the station's activities.
- Ethernet hardware (10M baud) was acquired (at no cost to the sponsor) to replace the less-powerful local-area network hardware (1M baud). Our station message protocol has been implemented on the Ethernet, connecting a VAX-11/750, the VAX-11/730, and numerous station module computers.
- A triangulation-range finder system (White Scanner) was acquired, debugged, and interfaced to the VAX-11/750. This system was utilized in the reported range-sensing research.
- An image processor (Imaging Technology IP-512) was acquired and interfaced to the network. Its purpose is to reduce the time it takes to convolve an image--specifically aiding in the computation

* By Randall C. Smith

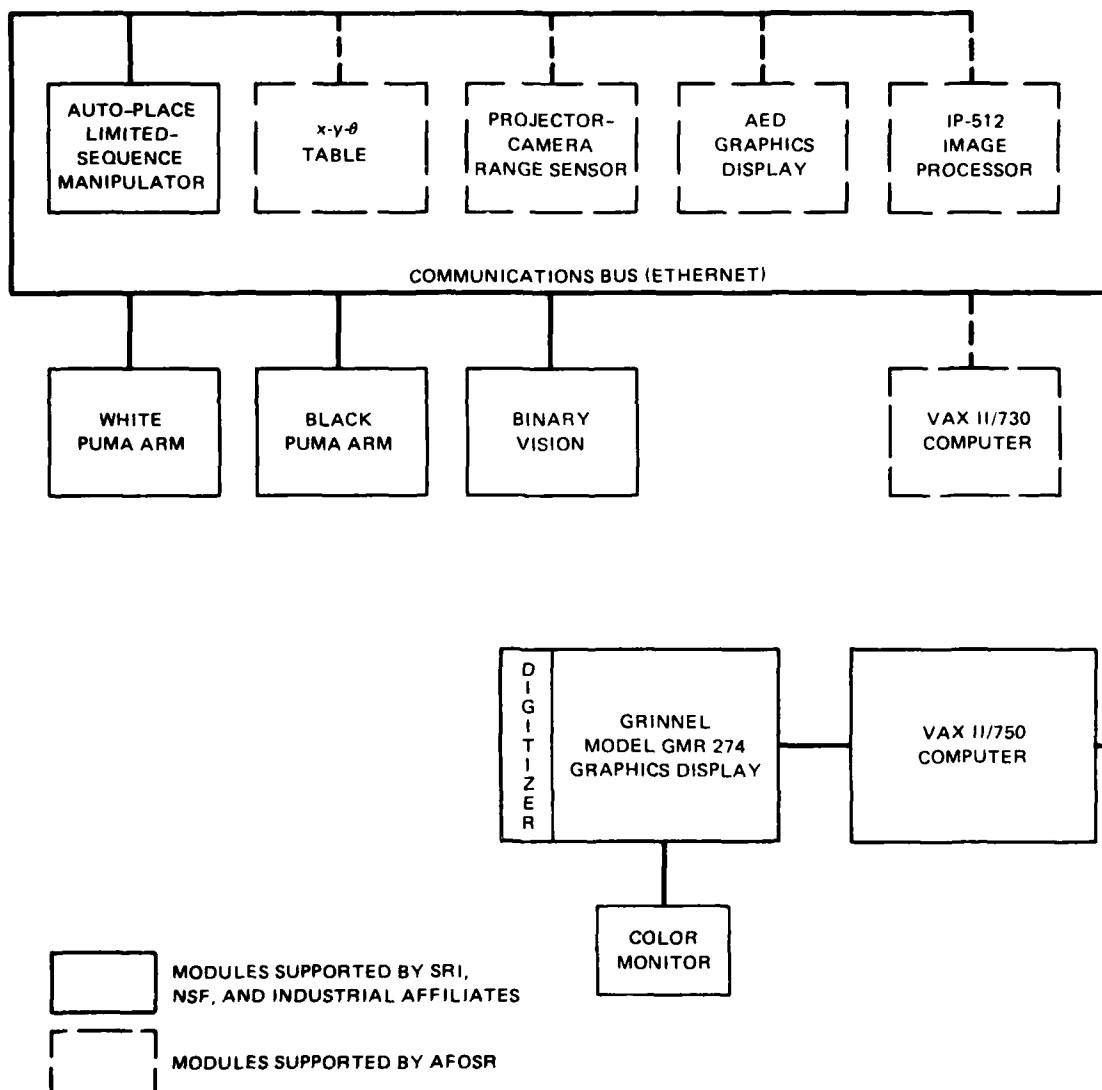


FIGURE 24 ASSEMBLY STATION BLOCK DIAGRAM

of zero-crossings, which now take many minutes on the VAX 11/750. Convolution software is under development.

- An existing $x-y-\theta$ table was upgraded and interfaced to the network.
- A 512 x 512 color graphics display (AED) was acquired and interfaced to the VAX-11/750 to support the sensing and collision-avoidance research. This display will shortly be connected to the network.
- Control over the Unimation PUMA 560 robots (two) was improved, by implementing new functions to take advantage of the ALTER mode of operation available in upgraded PUMAs. ALTER allows an external computer to change the PUMA's position and orientation 40 times each second.

The station was used for the collision-avoidance work described in Section V. The results of the collision-avoidance algorithms were demonstrated with the real arms, controlled over the local network. These collision-avoidance algorithms will soon be integrated into the station.

In another demonstration, a PUMA was controlled (using ALTER) across the network from a joy-stick on another machine, at the maximum 40 Hz rate that the PUMA controller allows, to demonstrate the capacity and flexibility of message-based control over the network, and to get some experience with ALTER.

Appendix A
FORMULAE FOR PRODUCT OF TWO NOMINAL
TRANSFORMATIONS

PRECEDING PAGE BLANK-NOT FILMED

Appendix A

FORMULAE FOR PRODUCT OF TWO NOMINAL TRANSFORMATIONS

$$x_g = x_2 * nx_1 + y_2 * ox_1 + z_2 * ax_1 + x_1$$

$$y_g = x_2 * ny_1 + y_2 * oy_1 + z_2 * ay_1 + y_1$$

$$z_g = x_2 * nz_1 + y_2 * oz_1 + z_2 * az_1 + z_1$$

$$\phi_g = \text{atan2}(ay_g, ax_g)$$

$$\theta_g = \text{atan2}(ax_g * \cos(\phi_g) + ay_g * \sin(\phi_g), az_g)$$

$$\psi_g = \text{atan2}(-nx_g * \sin(\phi_g) + ny_g * \cos(\phi_g), \\ -ox_g * \sin(\phi_g) + oy_g * \cos(\phi_g))$$

$$nx_1 = \cos(\phi_1) * \cos(\theta_1) * \cos(\psi_1) - \sin(\phi_1) * \sin(\psi_1)$$

$$ny_1 = \sin(\phi_1) * \cos(\theta_1) * \cos(\psi_1) + \cos(\phi_1) * \sin(\psi_1)$$

$$nz_1 = -\sin(\theta_1) * \cos(\psi_1)$$

$$ox_1 = -\cos(\phi_1) * \cos(\theta_1) * \sin(\psi_1) - \sin(\phi_1) * \cos(\psi_1)$$

$$oy_1 = -\sin(\phi_1) * \cos(\theta_1) * \sin(\psi_1) + \cos(\phi_1) * \cos(\psi_1)$$

$$oz_1 = \sin(\theta_1) * \sin(\psi_1)$$

$$ax_1 = \cos(\phi_1) * \sin(\theta_1)$$

$$ay_1 = \sin(\phi_1) * \sin(\theta_1)$$

$$az_1 = \cos(\theta_1) .$$

Formulae for $nx_2 \dots az_2$ correspond to the $nx_1 \dots az_1$ formulae above, but with substitution of variables subscripted by 2.

$$nx_3 = nx_1 * nx_2 + ox_1 * ny_2 + ax_1 * nz_2$$

$$ny_3 = ny_1 * nx_2 + oy_1 * ny_2 + ay_1 * nz_2$$

$$ox_3 = nx_1 * ox_2 + ox_1 * oy_2 + ax_1 * oz_2$$

$$oy_3 = ny_1 * ox_2 + oy_1 * oy_2 + ay_1 * oz_2$$

$$ax_3 = nx_1 * ax_2 + ox_1 * ay_2 + ax_1 * az_2$$

$$ay_3 = ny_1 * ax_2 + oy_1 * ay_2 + ay_1 * az_2$$

$$az_3 = nx_1 * ax_2 + oz_1 * ay_2 + az_1 * az_2$$

Appendix B
FUZZY TRANSFORMATION PRODUCT:
USING GAUSS'S LAW

Appendix B

FUZZY TRANSFORMATION PRODUCT:

USING GAUSS'S LAW

Compute: $FT_3 = FT_1 * FT_2$

Nominals: by formulas of Appendix A

Variances: computed for six components $(x_g, y_g, z_g, \phi_g, \theta_g, \psi_g)$

Random variables (v_i) :

$v_1 - v_6$ $x_1, y_1, z_1, \phi_1, \theta_1, \psi_1$ from FT_1

$v_7 - v_{12}$ $x_2, y_2, z_2, \phi_2, \theta_2, \psi_2$ from FT_2

General form:

$$\sigma_f^2 \approx g(f) = C_1^2 * \sigma^2(x_1) + \dots + C_{12}^2 * \sigma^2(\phi_2)$$

$$C_1 = \partial f / \partial v_1 = \partial f / \partial x_1$$

$$C_2 = \partial f / \partial v_2 \dots, C_{12} = \partial f / \partial v_{12}$$

Translational Variances

$$\sigma_{x3}^2 \approx g(\text{FNX3}(x_1, \phi_1, \theta_1, \psi_1, x_2, y_2, z_2))$$

$$\begin{aligned} C_1 &= 1, \quad C_2 = 0, \quad C_3 = 0, \\ C_4 &= -ny_1 * x_2 - oy_1 * y_2 - ay_1 * z_2, \\ C_5 &= \cos(\phi_1) * (nz_1 * x_2 + oz_1 * y_2 + az_1 * z_2), \\ C_6 &= ox_1 * x_2 - nx_1 * y_2, \\ C_7 &= nx_1, \quad C_8 = ox_1, \quad C_9 = ax_1, \\ C_{10} &= 0, \quad C_{11} = 0, \quad C_{12} = 0. \end{aligned}$$

$$\sigma_{y3}^2 \approx g(\text{FNY3}(y_1, \phi_1, \theta_1, \psi_1, x_2, y_2, z_2))$$

$$\begin{aligned} C_1 &= 0, \quad C_2 = 1, \quad C_3 = 0, \\ C_4 &= nx_1 * x_2 + ox_1 * y_2 + ax_1 * z_2, \\ C_5 &= \sin(\phi_1) * (nz_1 * x_2 + oz_1 * y_2 + az_1 * z_2), \\ C_6 &= oy_1 * x_2 - ny_1 * y_2, \\ C_7 &= ny_1, \quad C_8 = oy_1, \quad C_9 = ay_1, \\ C_{10} &= 0, \quad C_{11} = 0, \quad C_{12} = 0. \end{aligned}$$

$$\sigma_{z3}^2 \approx g(\text{FNZ3}(z_1, \phi_1, \theta_1, \psi_1, x_2, y_2, z_2))$$

$$\begin{aligned} C_1 &= 0, \quad C_2 = 0, \quad C_3 = 1, \\ C_4 &= 0, \\ C_5 &= -\cos(\psi_1) * az_1 * x_2 + \sin(\psi_1) * az_1 * y_2 - \sin(\theta_1) * z_2, \\ C_6 &= oz_1 * x_2 - nz_1 * y_2, \\ C_7 &= nz_1, \quad C_8 = oz_1, \quad C_9 = az_1, \\ C_{10} &= 0, \quad C_{11} = 0, \quad C_{12} = 0. \end{aligned}$$

ϕ_3 Variance

$$\sigma_{\theta_3}^2 \approx g(\text{FNPHI3}(\phi_1, \theta_1, \psi_1, \phi_2, \theta_2, \psi_2))$$

C_i have general form:

$$C_i = \frac{A_i * ax_3 - B_i * ay_3}{(ay_3)^2 + (ax_3)^2}$$

Note: since $ay_3^2 + ax_3^2 + az_3^2 = 1$ (its a unit vector)

$$\begin{aligned} \text{then } ay_3^2 + ax_3^2 \\ &= 1 - az_3^2 \\ &= 1 - (\cos(\theta_3))^2 \\ &= (\sin(\theta_3))^2 \end{aligned}$$

Thus, when $\theta_3 = n*\pi$ the correct value of C_i is 0. Values of θ_3 near $n*\pi$ cause trouble, even though ax_3 and ay_3 are themselves becoming very small.

$$C_4 = 1: \quad A_4 = ax_3, B_4 = ay_3,$$

$$C_5: \quad A_5 = \sin(\phi_1) * az_3, B_5 = \cos(\phi_1) * az_3$$

$$\begin{aligned} C_6: \quad A_6 &= ax_2 * oy_1 - ay_2 * ny_1 \\ B_6 &= ax_2 * ox_1 - ay_2 * nx_1 \end{aligned}$$

$$C_{10} = C_6,$$

$$C_{11}: \quad A_{11} = \cos(\phi_2) * az_2 * ny_1 + \\ \sin(\phi_2) * az_2 * oy_1 - \\ \sin(\theta_2) * ay_1$$

$$B_{11} = \cos(\phi_2) * az_2 * nx_1 + \\ \sin(\phi_2) * az_2 * ox_1 - \\ \sin(\theta_2) * ax_1$$

$$C_{12} = 0: \quad A_{12} = B_{12} = 0 \quad .$$

θ_3 Variance

$$\text{var}(\theta_3) \approx g(\text{FNTHETA3}(\phi_1, \theta_1, \psi_1, \phi_2, \theta_2, \psi_2))$$

C_i have general form:

$$C_i = E_i * az_3 - F_i * G,$$

$$\begin{aligned} \text{where } G &= ax_3 * \cos(\phi_3) + ay_3 * \sin(\phi_3), \\ \text{and } E_i &= dG/dv_i, \quad F_i = d(az_3)/dv_i. \end{aligned}$$

E_i as the general form

$$E_i = \frac{G * (ay_3 * A_i + ax_3 * B_i)}{ax_3^2 + ay_3^2}$$

where A_i, B_i are exactly the the same values found for $\sigma^2\phi_3$. Thus, the only new calculations are the F_i and E_i terms.

$$C_4 = 0: \quad A_4 = B_4 = 0, \quad F_4 = 0$$

$$C_5: \quad \text{if } ax_3^2 + ay_3^2 = 0 \text{ then } E_5 = 1.0$$

else compute E_5 from A_5, B_5

$$\begin{aligned} F_5 &= -ax_2 * \cos(\theta_1) * \cos(\psi_1) + ay_2 * \cos(\theta_1) * \sin(\psi_1) \\ &\quad - az_2 * \sin(\theta_1) \end{aligned}$$

$C_6:$ if $ax_3^2 + ay_3^2 = 0$ then $E_6 = 0.0$

 else compute E_6 from A_6, B_6

$$F_6 = ax_2 * oz_1 - ay_2 * nz_1$$

$$C_{10} = C_6$$

$C_{11}:$ if $ax_3^2 + ay_3^2 = 0$ then $E_{11} = 1.0$

 else compute E_{11} from A_{11}, B_{11}

$$F_{11} = \cos(\theta_2) * (\cos(\phi_2) * nz_1 + \sin(\phi_2) * oz_1) - \sin(\theta_2) * az_1$$

$$C_{12} = 0$$

ψ_3 Variance

$$\text{var}(\psi_3) = g(\text{FNPSI3}(\phi_1, \theta_1, \psi_1, \phi_2, \theta_2, \psi_2))$$

C_i have the general form:

$$M_i * J - N_i * K$$

$$\text{where } J = -ox_3 * \sin(\phi_3) + oy_3 * \cos(\phi_3),$$

$$K = -nx_3 * \sin(\phi_3) + ny_3 * \cos(\phi_3),$$

$$M_i = dK/dv_i, \quad N_i = dJ/dv_i$$

M_i have general form: (CPHI_i means the C_i calculated for ϕ_3 variance)

$$\text{CPHI}_i * (-\sin(\phi_3) * ny_3 - \cos(\phi_3) * nx_3) +$$

$$\cos(\phi_3) * d(ny_3)/dv_i - \sin(\phi_3) * d(nx_3)/dv_i$$

N_i have general form:

$$\text{CPHI}_i * (-\sin(\phi_3) * oy_3 - \cos(\phi_3) * ox_3) +$$

$$\cos(\phi_3) * d(oy_3)/dv_i - \sin(\phi_3) * d(ox_3)/dv_i$$

$$C_4 = 0$$

$$C_5: M_5: d(ny_3)/d\theta_1 =$$

$$-nx_2 * \sin(\phi_1) * \sin(\theta_1) * \cos(\psi_1) +$$

$$ny_2 * \sin(\phi_1) * \sin(\theta_1) * \sin(\psi_1) +$$

$$nz_2 * \sin(\phi_1) * \cos(\theta_1)$$

$$d(nx_3)/d\theta_1 =$$

$$-nx_2 * \cos(\phi_1) * \sin(\theta_1) * \cos(\psi_1) +$$

$$ny_2 * \cos(\phi_1) * \sin(\theta_1) * \sin(\psi_1) +$$

$$nz_2 * \cos(\phi_1) * \cos(\theta_1)$$

$$N_5: d(oy_3)/d\theta_1 =$$

$$-ox_2 * \sin(\phi_1) * \sin(\theta_1) * \cos(\psi_1) +$$

$$oy_2 * \sin(\phi_1) * \sin(\theta_1) * \sin(\psi_1) +$$

$$oz_2 * \sin(\phi_1) * \cos(\theta_1)$$

$$d(ox_3)/d\theta_1 =$$

$$-ox_2 * \cos(\phi_1) * \sin(\theta_1) * \cos(\psi_1) +$$

$$oy_2 * \cos(\phi_1) * \sin(\theta_1) * \sin(\psi_1) +$$

$$oz_2 * \cos(\phi_1) * \cos(\theta_1)$$

$$C_6: M_6: d(ny_3)/d\psi_1 = nx_2 * oy_1 - ny_2 * ny_1$$

$$d(nx_3)/d\psi_1 = nx_2 * ox_1 - ny_2 * nx_1$$

$$N_6: d(oy_3)/d\psi_1 = ox_2 * oy_1 - oy_2 * ny_1$$

$$d(ox_3)/d\psi_1 = ox_2 * ox_1 - oy_2 * nx_1$$

$$C_{10} = C_6$$

$$C_{11}: M_{11}: d(ny_3)/d\theta_2 =$$

$$-ny_1 * \cos(\phi_2) * \sin(\theta_2) * \cos(\psi_2) -$$

$$oy_1 * \sin(\phi_2) * \sin(\theta_2) * \cos(\psi_2) -$$

$$ay_1 * \cos(\theta_2) * \cos(\psi_2)$$

$$d(nx_3)/d\theta_2 =$$

$$-nx_1 * \cos(\phi_2) * \sin(\theta_2) * \cos(\psi_2) -$$

$$ox_1 * \sin(\phi_2) * \sin(\theta_2) * \cos(\psi_2) -$$

$$ax_1 * \cos(\theta_2) * \cos(\psi_2)$$

$$N_{11}: \quad d(oy_3)/d\theta_2 =$$

$$ny_1 * \cos(\phi_2) * \sin(\theta_2) * \sin(\psi_2) +$$

$$oy_1 * \sin(\phi_2) * \sin(\theta_2) * \sin(\psi_2) +$$

$$ay_1 * \cos(\theta_2) * \sin(\psi_2)$$

$$d(ox_3)/d\theta_2 =$$

$$nx_1 * \cos(\phi_2) * \sin(\theta_2) * \sin(\psi_2) +$$

$$ox_1 * \sin(\phi_2) * \sin(\theta_2) * \sin(\psi_2) +$$

$$ax_1 * \cos(\theta_2) * \sin(\psi_2)$$

$$C_{12} = 1$$

Appendix C

**FUZZY TRANSFORMATION PRODUCT:
USING SPHERICAL GEOMETRY MODEL**

PRECEDING PAGE BLANK-NOT FILMED

Appendix C
FUZZY TRANSFORMATION PRODUCT:
USING SPHERICAL GEOMETRY MODEL

Compute: $FT_3 = FT_1 * FT_2$

Nominals: by formulae of Appendix A

Variances: computed for three components (ϕ_3, θ_3, ψ_3)
use Gauss's Law formulae for x_3, y_3, z_3 variance

Formulae below use computations for sides and angles of right spherical triangles,
as follows:

$\text{atan}(\tan(\text{hypotenuse}) * \cos(\text{angle}))$ is side length adjacent to angle

$\text{asin}(\sin(\text{hypotenuse}) * \sin(\text{angle}))$ is side length opposite to angle

$\text{atan}(\tan(\text{angle}) * \sin(\text{adjside}))$ is side length opposite to angle

$\text{atan2}(\tan(\text{opposite}), \sin(\text{adjside}))$ is angle opposite to opposite

Refer to Figure C-1 for interpretation of these quantities used below:

$$\begin{aligned}
\eta &= \psi_3 - \psi_2 \\
\beta &= \phi_3 - \phi_1 \\
\rho &= \sqrt{\text{var}(\psi_1) + \text{var}(\phi_2)} \\
S &= \text{atan}(\tan(\rho) * \sin(\theta_2)) \\
S_2 &= \text{atan}(\tan(\rho) * \sin(\theta_1)) \\
t_1 &= \sigma_{\theta 1} \\
t_2 &= \sigma_{\theta 2} \\
\cos(\alpha) &= \sin(.5 * \pi - \theta_3) * \sin(\beta) \\
\cos(\delta) &= \sin(.5 * \pi - \theta_3) * \sin(\eta)
\end{aligned}$$

θ_3 Variance

$$\begin{aligned}
a &= \text{atan}(\tan(t_2) * \cos(\eta)) \\
a_1 &= \text{asin}(\sin(S) * \sin(\eta)) \\
a_2 &= \text{atan}(\tan(t_1) * \cos(\beta))
\end{aligned}$$

$$\text{var}(\theta_3) \approx a^2 + a_1^2 + a_2^2$$

ϕ_3 Variance

$$\begin{aligned}
b &= \text{asin}(\sin(t_2) * \sin(\eta)) \\
b_1 &= \text{atan}(\tan(S) * \cos(\eta)) \\
b_4 &= \text{atan}(\tan(t_1) * \cos(\alpha)) \\
\text{tmp} &= \sqrt{b^2 + b_1^2 + b_4^2}
\end{aligned}$$

$$\text{var}(\phi_3) = (\text{atan2}(\tan(\text{tmp}), \sin(\theta_3)))^2 + \text{var}(\phi_1)$$

ψ_3 Variance

$$b_2 = \text{asin}(\sin(t_1) * \sin(\beta))$$

$$b_3 = \text{atan}(\tan(S_2) * \cos(\beta))$$

$$b_5 = \text{atan}(\tan(t_2) * \cos(\delta))$$

$$\text{tmp} = \sqrt{b_2^2 + b_3^2 + b_5^2}$$

$$\text{var}(\psi_3) = (\text{atan2}(\tan(\text{tmp}), \sin(\theta_3)))^2 + \text{var}(\psi_2)$$

$$\text{If } \sin(\theta_1) = \sin(\theta_2) = 0, \text{ then } \text{var}(\psi_3) = \text{var}(\psi_2) + \rho^2.$$

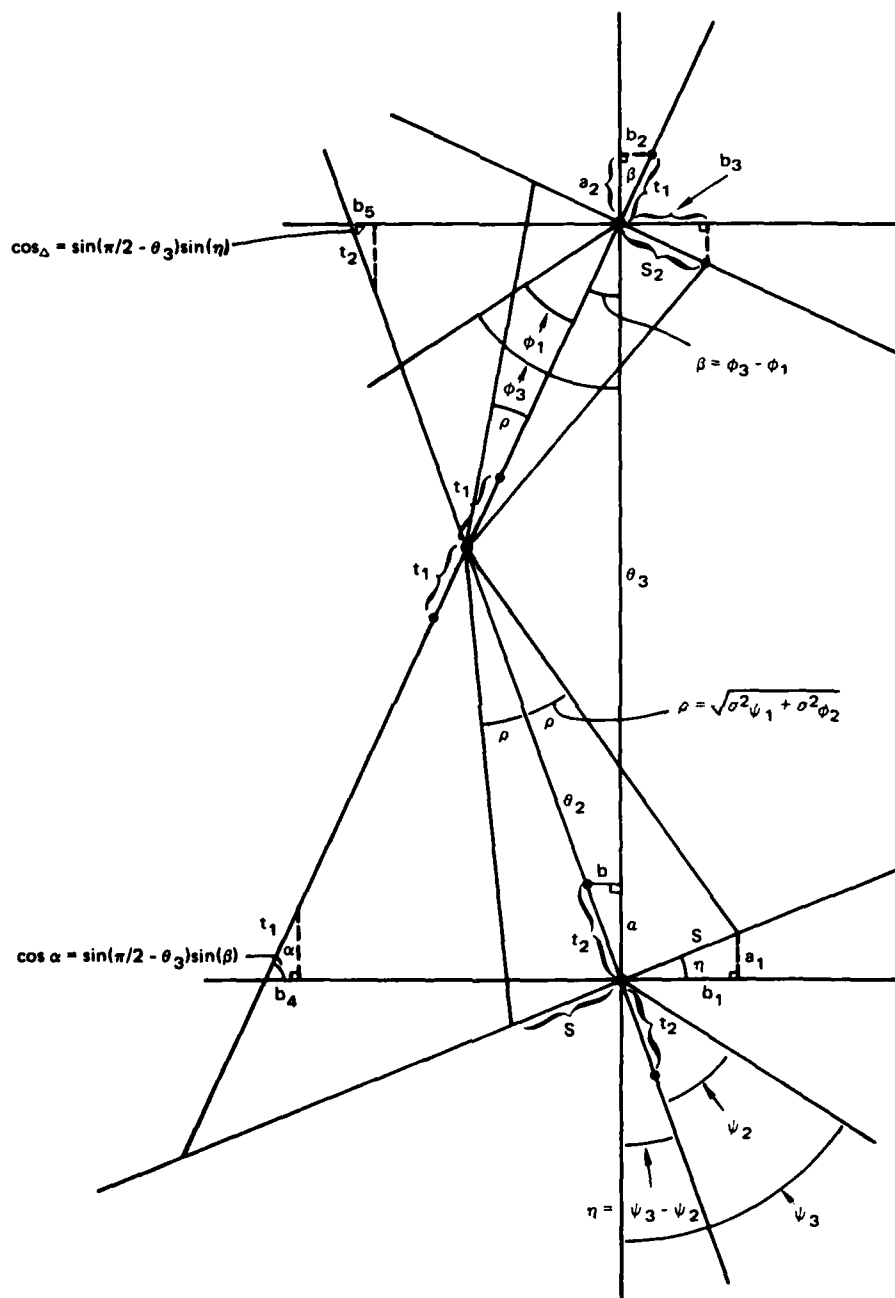


FIGURE C-1 COMPONENTS IN THE ESTIMATION OF ANGULAR VARIANCES

Appendix D
ALGORITHMS FOR COLLISION AVOIDANCE

Appendix D

ALGORITHMS FOR COLLISION AVOIDANCE

I. Attraction Towards Goal

1. Joint space versus Cartesian space
2. Infinite versus finite attraction at goal
3. Increasing towards goal versus constant over space
4. Increasing with time versus constant over time.

II. Obstacle Fields

1. $\| F \| = \iiint \nabla f(\bar{x}) d\bar{x} \dots$ or,
2. $\| F \| = e^{\| \bar{c}_2 - \bar{c}_1 \| - ((r_1 + r_2) + \text{cushion})}$

where

$$F = \| F \| (\bar{c}_2 - \bar{c}_1) / (\| \bar{c}_2 - \bar{c}_1 \|)$$

III. Influences

$$F_{\text{total}} = p * F_{\text{goal}} + (1-p) * F_{\text{obstacles}}, \quad 0 \leq p \leq 1.$$

IV. Resulting Motion

1. $\tau = J^T * F$

- a. Full dynamic simulation
- b. Hand mass only--massless arm
- c. Unit hand mass only--massless arm
- d. Overriding damping: $w = k \tau$ assumption
- e. Sum of resulting torques, versus reflecting forces to the hand and computing with the sum of resulting forces.

2. $dQ = J^{-1} dP$

- a. Direct application
- b. Interactive forward approximation
- c. Sum of resulting dQ , versus reflecting dP to the hand and computing with the sum of resulting dP .

3. Arm Solution

- a. Sum of resulting ΔQ , versus reflecting ΔP to the hand and computing with the sum of the resulting ΔP .

Appendix E
PUBLICATIONS

Appendix E
PUBLICATIONS

1. Park, W.T., "State-Space Representations for Coordination of Multiple Manipulators," to be published in *Proc. of 14th International Symposium on Industrial Robotics*, Gothenburg, Sweden, October 2-4, 1984.
2. Myers, J.K., "RCODE: The Robotic Simulator with Collision Detection," (in preparation).
3. Smith, R.C., and Cheeseman, P., "Analysis of Location Uncertainty," (in preparation)

Appendix F
PERSONNEL

PRECEDING PAGE BLANK-NOT FILMED

Appendix F
PERSONNEL

Principal Investigators:

Randall C. Smith, Research Engineer, Robotics Laboratory
David Nitzan, Director, Robotics Laboratory

Task 1: Analysis of Location Uncertainty

Randall C. Smith, Research Engineer

Task 2: Acquisition and Analysis of Range Data

Robert C. Bolles, Senior Computer Scientist
James H. Herson, Computer Scientist

Task 3: Characterization of Feature Detectors

Robert C. Bolles, Senior Computer Scientist
James H. Herson, Computer Scientist

Task 4: Multiarm Collision Avoidance

John K. Myers, Research Engineer

Task 5: Coordination of Multiple Manipulators

William T. Park, Senior Research Engineer

Task 6: Extension of a Programmable Assembly Station

Randall C. Smith, Robert C. Bolles, James H. Herson,
John K. Myers, and William T. Park

Appendix G
DISTRIBUTION LIST

PRECEDING PAGE BLANK-NOT FILMED

Appendix G
DISTRIBUTION LIST

AFOSR

Dr. Alan H. Rosenstein
Program Manager
Electronic and Material Sciences
Air Force Office of Scientific Research
Bolling Air Force Base, DC 20332

Lt Col Harry V. Winsor
Acting Director
Electronic and Material Sciences
Air Force Office of Scientific Research
Bolling Air Force Base, DC 20332

Air Force

Lt Michael F. Hitchcock
Project Leader
Manufacturing Sciences Program
Air Force Materials Laboratory (AFWAL/ML)
Wright-Patterson AFB, OH 45433

Dr. Vincent J. Russo
Director
Manufacturing Sciences Program
Air Force Materials Laboratory (AFWAL/ML)
Wright-Patterson AFB, OH 45433

Ted Brandewie
Manufacturing Sciences Program
Air Force Materials Laboratory (AFWAL/ML)
Wright-Patterson AFB, OH 45433

DARPA

Dr. William E. Isler
Program Manager, Robotic Systems
Systems Sciences Division
Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

Dr. Clinton Kelly
Director, Defense Sciences Office
Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

Army

Mr. Timothy Evans
U.S. Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27701

Navy

LCDR Bart Everett
Special Assistant for Robotics
Naval Sea Systems Command, C90-M
Washington, DC 20362

Dr. Paul Schneck
Information Sciences Program
Office of Naval Research, Code 433
800 N. Quincy
Arlington, VA 22217

NSF

Mr. Norman Caplan
Program Manager, Elec., Comp., and System Eng.
National Science Foundation
1800 G Street, N.W.
Washington, DC 20550

Dr. William M. Spurgeon
Program Director, Production Research
Div. of Mechanical Eng. & Applied Mech.
National Science Foundation, Room 1108
1800 G. Street, N.W.
Washington, DC 20550

NBS

Dr. James Albus
Division Chief, Industrial Systems Div.
Center for Manufacturing Engineering
National Bureau of Standards
Bldg. 220, Room A123
Washington, DC 20234

Stanford University

Dr. Thomas O. Binford
Department of Computer Science
Stanford University
Stanford, CA 94305

Dr. Robert H. Cannon
Stanford University
250 Durand Bldg.
Stanford, CA 94305

University of Michigan
Dean Daniel Atkins
University of Michigan
Chrysler Center
Ann Arbor, Michigan 48109

Brigham Young University
Del Allen
Brigham Young University
Provo, Utah 84602

References

1. Albus, J., *Brains, Behavior, & Robotics*, BYTE Books, Subsidiary of McGraw-Hill, Peterborough, 1981.
2. Brooks, R.A., "Symbolic Error Analysis and Robot Planning," *International Journal of Robotics Research*, Vol. 1, No. 4 (Winter 1982).
3. Brooks, R.A., and T. Lozano-Perez, "A Subdivision Algorithm in Configuration Space for Findpath with Rotation," AIM-684, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts (December 1982).
4. Brooks, R.A., "Solving the Find-Path Problem by Representing Free Space as Generalized Cones," AIM-674, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts (May 1982).
5. Brooks, R.A., "Planning Collision-Free Motions for Pick-and-Place Operations," *Int. Journal of Robotics Research*, Vol. 2, No. 4 (Winter 1983) 19-44.
6. Howden, W.D., "The Sofa Problem," *Computer Journal*, Vol. 11 (November 1968) 299-301.
7. Khatib, O., and J.-F. LeMaitre, "Dynamic Control of Manipulators Operating in a Complex Environment," *Proc. 3rd CISM-IFTOMM Symp. on the Theory and Practice of Robots and Manipulators*, pp. 267-282, Udine, Italy (September 1978).
8. Khatib, O., "Dynamic Control for Manipulators in Operational Space," *6th IFTOMM Congress on Theory of Machines and Mechanisms*, New Delhi, India (December 15-20, 1983).
9. Kirkpatrick, S., C.D. Gelatt, Jr., and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598 (May 13, 1983) 671-680.

10. Lewis, R.A., "Autonomous Manipulation on a Robot: Summary of Manipulator Software Functions," Technical Report 33-679, Jet Propulsion Laboratory, Pasadena, California (March 1974).
11. Lozano-Perez, T., "The Design of a Mechanical Assembly System," TR-397, Massachusetts Institute of Technology, Cambridge, Massachusetts (December 1976).
12. Lozano-Perez, T., and M.A. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Communications of ACM*, pp. 560-570, ACM 22 (October 1979). Published as Volume 10.
13. Marr, D., and E. Hildreth, "Theory of Edge Detection," *Proc. R. Soc. Lond., Vol. B-207*, pp. 187-217, London, Great Britain (1980).
14. Myers, J.K., *A Supervisory Collision-Avoidance System for Robot Controllers*, Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, Pennsylvania, December 1981.
15. Myers, J.K., and G.J. Agin, "A Supervisory Collision-Avoidance System for Robot Controllers," *Robotics Research and Advanced Applications*, American Society of Mechanical Engineers, W.J. Book, ed., New York, New York, 1983, pp. 225-232.
16. Myers, J.K., "RCODE: The Robotic Simulator with Collision Detection" to be published
17. Paul, R.P., *Robot Manipulators*, MIT Press, Cambridge, Massachusetts, 1981.
18. Pavlidis, T., and S.L. Horowitz, "Segmentation of Plane Curves," *IEEE Transactions on Computers*, C-23, 860-870 (August 1974).
19. Pavlidis, T., "Curve Fitting as a Pattern Recognition Problem," *6th Int. Conf. on Pattern Recognition*, Munich, Germany (October 1982).
20. Reif, J., "On the Movers' Problem," *Proc. 20th Annual IEEE Symp. Foundation of Computer Science*, pp. 421-427 (1979).

21. Schwartz, J.T., and M. Sharir, "On the Piano Movers' Problem II: General Techniques for Computing Topological Properties of Real Manifolds," 41, Department of Computer Science, Courant Institute, New York University, New York, New York (1982).
22. Smith, R.C., and D. Nitzan, "A Modular Programmable Assembly Station," *Proc. 13th Int. Symposium on Industrial Robots*, pp. 5.53-5.75, Chicago, Illinois (April 18-21, 1983).
23. Taylor, R.H., *A Synthesis of Manipulation Control Programs from Task Level Specifications*, Ph.D. Thesis, Stanford University, Stanford, California, 1976.
24. Udupa, S.M., "Collision Detection and Avoidance in Computer-Controlled Manipulators," *Proc. 5th Int. Joint Conf. on Artificial Intelligence*, Cambridge, Massachusetts (1977).
25. Wahl, F.M., and K.Y. Wong, "An Efficient Method for Running a Constrained Run Length Algorithm (CRLA) in Vertical and Horizontal Direction on Binary Image Data," RJ3483, IBM Research Division, San Jose, California (March 1982).
26. Wallace, R., . Personal communication.
27. Yau, M.M., and S.N. Srihari, "A Hierarchical Data Structure for Multidimensional Digital Images," *Communications of ACM*, 26, 7, pp. 504-515 (July 1983).